CURSO PRÁTICO 69 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

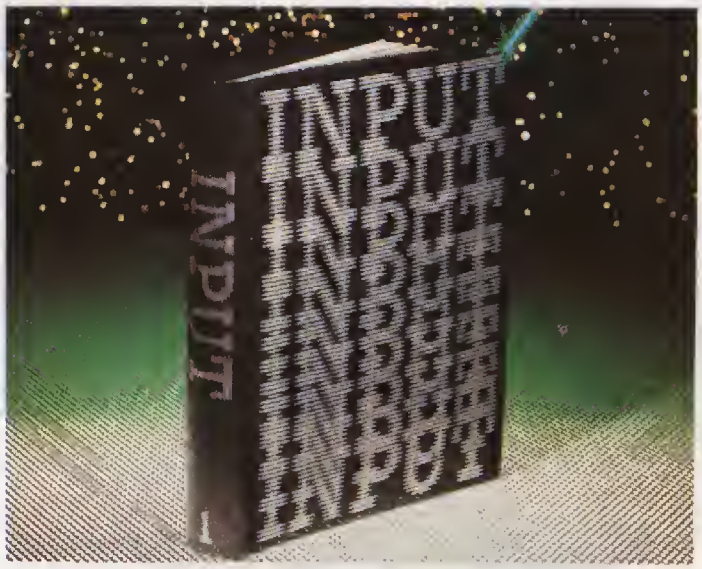
PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

Cz$ 50,00

INPUT

Vol. 5 N.º 69

NESTE NÚMERO



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

EDITOR
RICHARD CIVITA

NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues, Pierluigi Bracco, Plácido Nicoleto, Roberto Silveira, Shozi Ikeda, Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos: Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena, Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista

Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi

Colaboradores
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado: Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez, Menahen M. Politi, Renê C. X. Santos, Stélio Alves Campos

Esta obra foi composta na AM Produções Gráficas Ltda e impressa na Divisão Gráfica da Editora Abril S.A.

DOMINE O VÍDEO DO MSX

- TELA DE QUARENTA COLUNAS
- VPEEK E VPOKE
- FORMATO DAS LETRAS
- TABELA DE PADRÕES
- NOVOS CARACTERES

O MSX difere dos demais micros por possuir um microprocessador e uma memória de dezesseis kbytes dedicados só ao controle da tela. Desvende os segredos da VRAM e sinta a diferença!

Muitos usuários dos micros MSX enfrentam dificuldades em utilizar comandos como **BASE, VPEEK** e **VPOKE.** Para melhor aproveitar esses recursos da linguagem BASIC, temos inicialmente que entender como é organizada a memória de vídeo — VRAM —, um item desprezado pela maioria dos manuais.

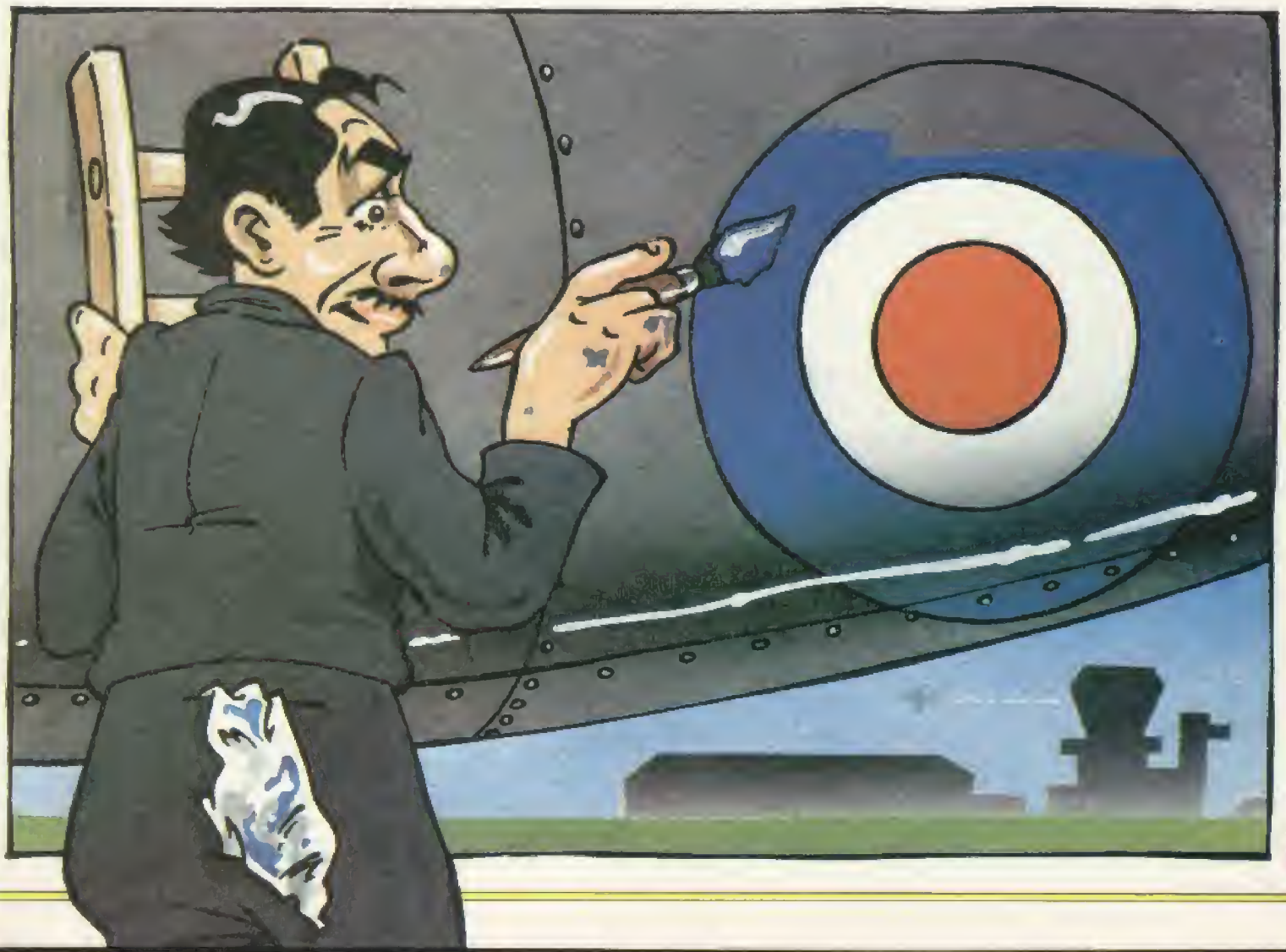
O MSX possui um chip independente da unidade central de processamento (CPU) destinado exclusivamente ao controle da tela. Conhecido como VDP (*Video Display Processor*), ele gera imagens a partir de dados armazenados nos dezesseis kbytes de uma memória independente da RAM (memória de acesso direto, onde ficam armazenados o programa BASIC e as variáveis). Dessa forma, a utilização de gráficos multicoloridos de alta resolução não compromete a quantidade de memória disponível, como acontece com outros micros.

TELA DE QUARENTA COLUNAS

Ao ligarmos um microcomputador da linha MSX, observamos uma tela de textos em que cada linha acomoda, teoricamente, quarenta caracteres. Se isso não ocorrer em seu micro, digite o comando **SCREEN 0.**

A tela — quer contenha textos, listagens de programas ou símbolos gráficos — é simplesmente uma reprodução da parte da VRAM denominada *Tabela de Nomes* (TN). Essa área da memória de vídeo possui 960 bytes de comprimento, correspondentes aos 960 caracteres que podem ser mostrados na tela simultaneamente (são 24 linhas com quarenta caracteres cada uma).

Cada uma dessas posições pode conter um número qualquer entre 0 e 255, número que é interpretado pelo VDP como o código ASCII (sigla de *American Standard Code for Information Interchange*) de um caractere.



VPEEK E VPOKE

Em qualquer micro, o comando **POKE** do BASIC nos permite armazenar um número entre 0 e 255 numa posição da RAM. Da mesma forma, o conteúdo de qualquer endereço da RAM ou da ROM é obtido com o comando **PEEK**.

Como a VRAM não depende da RAM, necessitamos de comandos que nos possibilitem ler ou escrever bytes diretamente na memória de vídeo, ou seja, **VPEEK** e **VPOKE**. Experimente essa nova forma de escrever na tela:

```
10 CLS
20 VPOKE 417,77
30 VPOKE 419,83
40 VPOKE 421,88
50 END
```

Esse programa coloca diretamente nas posições 417, 419 e 421 os códigos das letras M, S e X. O VDP transforma o código da Tabela de Nomes no caractere correspondente, que é imediatamente reproduzido na tela.

Para conferir, digite:

```
PRINT VPEEK (417)
PRINT VPEEK (419)
PRINT VPEEK (421)
```

Estes comandos diretos recuperam os códigos das letras M, S e X diretamente da Tabela de Nomes. Se você não gosta muito de códigos, obtenha as letras correspondentes utilizando:

```
PRINT CHR$(VPEEK (417))
PRINT CHR$(VPEEK (419))
PRINT CHR$(VPEEK (421))
```

Depois de familiarizado com o funcionamento desses dois comandos, o usuário poderá ter acesso às quarenta colunas da tela (o que não é possível por intermédio do comando **PRINT**). Muitos leitores já devem ter observado que o texto começa a ser escrito a partir da segunda coluna, reduzindo, portanto, o espaço total em uma coluna.

Para ilustrar isso, note que

```
LOCATE 0,10:PRINT "*"
```

imprime um asterisco na segunda coluna, deixando um espaço vazio à esquerda, enquanto os comandos

```
LOCATE 38,10:PRINT "*"
```

e

```
LOCATE 39,11:PRINT "*"
```

imprimem asteriscos na mesma coluna.

Use o seguinte programa para ocupar a primeira coluna da tela:

```
10 SCREEN 0:KEY OFF
20 FOR I=0 TO 959 STEP 40
30 VPOKE BASE(0)+I,207
40 NEXT
```

A linha 10 seleciona a tela de textos e apaga os rótulos das teclas de função da parte inferior do vídeo. Um laço **FOR..NEXT** entre as linhas 20 e 40 coloca várias vezes o caractere de código 207 na primeira posição de cada linha. Deve-se ressaltar que, apesar de termos usado a instrução **BASE(0)** na linha 30, ela é perfeitamente dispensável, uma vez que seu valor é zero.

COMO SE DESENHAM AS LETRAS

Vimos até agora que, ao se colocar um número dentro da Tabela de Nomes, o VDP imediatamente o interpreta como o código ASCII de uma letra ou de um símbolo. Como o processador sabe o formato de cada uma das letras do alfabeto?

O leitor já deve ter notado que as letras que aparecem no vídeo da TV são compostas por pequenos pontos. Cada caractere tem seu perfil — ou padrão — desenhado em um quadrado com oito pontos de lado. Este padrão é codificado em números binários (formados por zeros e uns), a fim de que o computador possa compreendê-lo. Se fizermos com que os "uns" correspondam aos pontos acesos e os "zeros", aos apagados, poderemos codificar todos os perfis. Como um número binário entre 0 e 255 tem no máximo oito algarismos, cada caractere será representado através de uma série de oito números binários menores que 255, ou seja, oito bytes.

Para que o VDP reconheça o formato dos caracteres, uma porção da VRAM é separada para armazenar os 2048 bytes necessários para representar os 256 símbolos existentes (256*8 = 2048). Esta parte da memória de vídeo é chamada de Tabela de Padrões e começa no endereço 2048 da VRAM.

BASE

Para que o usuário não confunda os endereços da RAM e da VRAM, o BASIC do MSX oferece a instrução **BASE**, que evita a memorização dos endereços e permite modificar a posição das tabelas. Assim, para a tela de textos de quarenta colunas, o endereço inicial da Tabela de Nomes está em **BASE (0)** e o da Tabela de Padrões em **BASE (2)**. Confira:

```
PRINT BASE(0)
```

e

```
PRINT BASE(2)
```

TABELA DE PADRÕES

Visando a uma melhor compreensão da Tabela de Padrões, mostraremos um pequeno programa que nos facilitará o acesso a essa região da VRAM.

Ao ser executado, o programa imprime na tela a porção da tabela que corresponde a um caractere. São impressos também os endereços (com conteúdos em números decimal e binário) e o caractere com seu código. Dessa forma, o leitor entenderá como os números binários codificam o formato dos caracteres.

```
10 Z$="00000000"
20 SCREEN 0:KEY OFF
30 J=520
40 GOSUB100
50 K$=INKEY$:IF K$="" THEN 50
60 IF K$=CHR$(31) AND J<2040 TH
EN J=J+8
70 IF K$=CHR$(30) AND J>7 THEN
J=J-8
```

```
80 GOTO 40
100 LOCATE 12,1:PRINT "CARACTER
: "
110 VPOKE 64,J/8
120 LOCATE 10,3:PRINT "Código A
SCII: ";J/8
130 LOCATE 1,6:PRINT "Endereço"
;TAB(14);"Conteúdo";TAB(27);"Co
nteúdo"
140 PRINT:PRINT TAB(3);"VRAM";T
AB(14);"Decimal";TAB(27);"Binár
io"
150 PRINT
160 FOR I=J TO J+7
170 PRINT TAB(2);BASE(2)+I;TAB(
15);VPEEK(BASE(2)+I);TAB(27);RI
GHT$(ZS+BIN$(VPEEK(BASE(2)+I)),
8)
180 NEXT
190 RETURN
```

A linha 10 cria uma variável auxiliar que permite a conversão decimal/binário. A linha 20 seleciona a tela de quarenta colunas e apaga o rodapé com as teclas de função.

A variável **J** controla a porção da Tabela de Padrões mostrada na tela, definindo o caractere que tem seu padrão exibido. Seu valor inicial, determinado na linha 30, faz com que a primeira letra impressa seja o 'A' — cujo código é 65 (65*8 = 520). A linha 40 chama a sub-rotina que imprime a porção da tabela que nos interessa. As linhas 50 e 80 permitem avançar ou retroceder dentro da Tabela de Padrões usando as teclas de controle do cursor.

A sub-rotina que cuida da impressão dos dados na tela vai da linha 100 à 190. As linhas 100 e 110 imprimem o caractere em questão; a linha 120 imprime seu código e as linhas 130 a 150 encarregam-se do cabeçalho.

O laço **FOR..NEXT**, da linha 160 à 180, imprime o conteúdo dos oito bytes que determinam o padrão do caractere e seus endereços na VRAM.

Depois de uma ligeira exploração da Tabela de Padrões, o leitor notará que os caracteres gráficos (aqueles que não são letras ou sinais de pontuação) surgem truncados no vídeo. A razão disso é a incapacidade do MSX de imprimir mais que 256 pontos horizontais. Essa limitação fica mais evidente quando precisamos de uma resolução de 320 pontos na horizontal para representar quarenta caracteres, cada um com oito pontos de largura. Assim, quando estamos em **SCREEN 0**, somente são mostrados os seis pontos — ou bits — mais à esquerda do padrão, suficientes para definir as letras mas não os caracteres gráficos, que muitas vezes utilizam até oito pontos.

UM NOVO CONJUNTO DE CARACTERES

O formato das letras é definido na VRAM, e nada nos impede de modificá-lo, já que dispomos do comando **VPOKE** para executar tal tarefa. Confira:

```
10 FOR I=65*8 TO 91*8-1
20 READ A:VPOKE BASE(2)+I,A
30 NEXT
40 FOR I=97*8 TO 123*8-1
50 READ A:VPOKE BASE(2)+I,A
60 NEXT
70 FOR I=48*8 TO 58*8-1
80 READ A:VPOKE BASE(2)+I,A
90 NEXT
100 END
1000 DATA 120,72,72,200,248,200
,200,0
1010 DATA 112,80,80,248,200,200
,240,0
1020 DATA 120,72,64,192,192,200
,248,0
1030 DATA 112,72,72,200,200,200
,240,0
1040 DATA 120,64,64,240,192,192
,248,0
1050 DATA 120,64,64,240,192,192
,192,0
1060 DATA 120,64,64,216,200,200
,248,0
1070 DATA 72,72,72,248,200,200,
200,0
1080 DATA 16,16,16,48,48,48,48,
0
1090 DATA 48,16,16,24,152,152,1
20,0
1100 DATA 72,72,80,224,208,200,
200,0
1110 DATA 64,64,64,192,192,192,
248,0
1120 DATA 68,108,84,196,196,196
,196,0
1130 DATA 72,104,104,216,216,21
6,200,0
1140 DATA 120,72,72,200,200,200
,248,0
1150 DATA 120,72,72,248,192,192
,192,0
1160 DATA 120,72,72,200,200,216
,248,4
1170 DATA 112,72,72,248,208,200
```

```
,200,0
1180 DATA 112,80,64,120,24,152,
248,0
1190 DATA 248,32,32,32,96,96,96
,0
1200 DATA 72,72,72,200,200,200,
248,0
1210 DATA 200,200,200,200,200,8
0,32,0
1220 DATA 196,196,212,212,212,4
0,40,0
1230 DATA 80,80,80,32,208,200,2
00,0
1240 DATA 136,136,136,80,48,48,
48,0
1250 DATA 120,72,16,224,192,200
,248,0
1260 DATA 0,0,112,16,240,208,24
8,0
1270 DATA 128,128,128,240,200,2
00,248,0
1280 DATA 0,0,112,64,192,192,24
8,0
1290 DATA 16,16,112,80,208,208,
248,0
1300 DATA 0,0,112,80,240,192,24
8,0
1310 DATA 48,80,64,64,240,192,1
92,0
1320 DATA 0,240,144,224,64,248,
200,248
1330 DATA 64,64,64,248,200,200,
200,0
1340 DATA 16,0,16,16,48,48,48,0
1350 DATA 16,0,16,16,24,152,152
,120
1360 DATA 0,64,72,80,224,208,20
0,0
1370 DATA 32,32,32,96,96,96,112
,0
1380 DATA 0,0,128,208,168,168,1
68,0
1390 DATA 0,0,64,112,200,200,20
0,0
1400 DATA 0,0,112,200,200,200,1
12,0
1410 DATA 0,0,120,72,200,248,19
2,192
1420 DATA 0,0,240,144,144,248,2
4,24
1430 DATA 0,0,80,104,192,192,19
2,0
1440 DATA 0,0,112,64,56,136,248
,0
1450 DATA 32,32,112,32,32,96,11
2,0
1460 DATA 0,0,72,72,200,200,248
,0
1470 DATA 0,0,200,200,200,80,32
,0
1480 DATA 0,0,136,168,168,168,8
0,0
1490 DATA 0,0,80,80,32,208,200,
0
1500 DATA 0,0,72,72,48,48,48,0
1510 DATA 0,0,120,8,112,96,120,
0
1520 DATA 120,72,72,216,232,200
,248,0
1530 DATA 32,96,32,48,48,48,48,
0
1540 DATA 120,72,8,248,192,200,
248,0
1550 DATA 112,16,16,120,24,152,
248,0
1560 DATA 144,144,144,248,24,24
,24,0
1570 DATA 240,128,128,248,24,24
,248,0
1580 DATA 112,64,64,248,200,200
,248,0
1590 DATA 224,32,32,48,48,48,48
,0
1600 DATA 112,80,80,248,152,152
,248,0
1610 DATA 240,144,144,248,24,24
,248,0
```

O conjunto de letras futuristas permanece à disposição do usuário até que o micro seja desligado ou receba um comando **SCREEN**.

O programa é composto de três laços **FOR...NEXT** e um grande número de linhas **DATA**, cada uma com oito bytes que definem o padrão dos símbolos.

O primeiro laço — que vai da linha 10 a 30 — modifica os padrões das letras maiúsculas (códigos ASCII de 65 a 90). O segundo laço — linhas 40 a 60 — cuida das minúsculas (códigos 97 a 122) e o terceiro — linhas 70 a 90 — dos números (48 a 57). Não incluímos caracteres acentuados para não estender muito as linhas **DATA**. Entretanto, se o leitor compreender bem o funcionamento do programa, não terá dificuldade em criá-los, completando assim o conjunto.

O programa que explora a Tabela de Padrões poderá ser aplicado para que se conheça mais detalhadamente os caracteres recém-criados.

Em contraste com o traço de vanguarda desses símbolos artificiais, criamos um segundo conjunto, dessa vez em letras de fôrma. As próximas linhas deverão ser acrescentadas ao programa anterior, formando um maior, que será completado no final do artigo.

```
5 RESTORE 1620
1620 DATA 16,40,72,72,120,72,13
2,0
1630 DATA 120,36,36,56,36,68,24
8,0
1640 DATA 24,36,64,64,64,68,56,
0
1650 DATA 120,36,36,36,36,68,24
8,0
1660 DATA 24,36,32,56,64,68,56,
0
1670 DATA 60,72,16,16,60,144,96
,0
1680 DATA 24,36,64,68,68,60,136
,112
1690 DATA 68,72,72,72,120,72,13
2,0
1700 DATA 8,24,40,8,8,16,224,0
1710 DATA 60,72,8,8,8,136,144,9
6
1720 DATA 68,72,80,96,80,72,132
,0
1730 DATA 32,64,64,64,64,68,248
,0
1740 DATA 40,84,84,84,84,84,132
,0
1750 DATA 68,68,100,84,76,68,13
2,0
1760 DATA 24,36,68,68,68,72,48,
0
1770 DATA 120,36,36,36,120,32,1
92,0
1780 DATA 24,36,68,68,116,72,52
,0
1790 DATA 120,36,36,36,120,40,1
96,0
1800 DATA 24,36,64,56,4,136,112
,0
1810 DATA 124,16,16,16,16,32,19
2,0
1820 DATA 72,72,72,72,72,72,52,
0
1838 DATA 76,72,72,72,72,80,32,
0
1840 DATA 68,84,84,84,84,84,40,
0
1850 DATA 68,36,40,16,40,72,132
,0
1860 DATA 68,36,36,36,24,16,224
,0
1870 DATA 60,68,8,124,32,68,248
,0
1880 DATA 0,0,56,72,136,136,116
,0
1890 DATA 32,64,64,120,68,68,24
8,0
1900 DATA 0,0,24,36,64,64,188,0
1910 DATA 4,4,28,36,68,68,188,0
1920 DATA 0,0,56,68,72,48,220,0
1930 DATA 8,20,32,32,32,252,32,
32
1940 DATA 0,0,60,68,68,184,8,11
2
1950 DATA 32,32,40,52,36,100,16
4,0
1960 DATA 16,0,16,16,16,48,204,
0
1970 DATA 8,0,24,40,72,140,8,11
2
1980 DATA 16,32,36,40,48,40,196
,0
1990 DATA 16,40,40,40,40,16,236
,0
2000 DATA 0,0,40,84,84,84,132,0
2010 DATA 0,0,88,100,68,68,132,
0
2020 DATA 0,0,28,36,92,68,184,0
2030 DATA 0,0,56,36,36,252,32,3
2
2040 DATA 0,0,56,68,68,188,4,4
2050 DATA 0,0,32,60,36,68,132,0
2060 DATA 0,32,48,40,36,68,152,
0
2070 DATA 36,16,124,16,16,48,20
4,0
2080 DATA 0,0,68,68,68,68,188,0
2090 DATA 0,0,76,72,72,80,160,0
2100 DATA 0,0,196,84,84,84,40,0
2110 DATA 0,0,68,40,16,40,196,0
2120 DATA 0,0,36,36,36,252,8,11
2
2130 DATA 0,0,124,4,56,64,252,0
2140 DATA 24,36,76,84,100,72,48
,0
2150 DATA 8,24,8,16,16,32,112,0
2160 DATA 56,68,36,8,48,100,88,
```

```
0
2170 DATA 24,36,4,24,68,68,56,0
2188 DATA 4,12,24,40,72,124,16,
0
2190 DATA 60,32,64,120,4,4,120,
0
2200 DATA 12,16,32,120,68,68,56
,0
2210 DATA 60,4,8,16,32,64,64,0
2220 DATA 56,68,68,56,68,68,56,
0
2230 DATA 24,36,68,68,56,8,112,
0
```

O programa utilizado é o mesmo, com alterações apenas nas linhas **DATA**. O comando **RESTORE 1620** faz com que o comando **READ** inicie a leitura a partir da linha **DATA** 1620.

MODIFICANDO O VALOR-BASE

A VRAM tem 16384 bytes de comprimento, dos quais utilizamos apenas 3008 quando estamos no modo **SCREEN 0** (2048 para a Tabela de Padrões e 960 para a de nomes). Esse espaço ocioso pode ser aproveitado para armazenar conjuntos de caracteres e cópias da tela. O segredo do processo consiste na atribuição de novos valores para as instruções **BASE(0)** e **BASE(2)**.

As linhas seguintes devem ser adicionadas aos dois programas anteriores, formando, assim, um terceiro. É necessário apagar as linhas 5 a 100 com **DELETE 5-100**. Além disso, o programa precisa ser gravado antes de ser testado, a fim de permitir correções posteriores, no caso de ter ocorrido qualquer erro de digitação.

```
10 SCREEN 3:SCREEN 0:KEY OFF
20 FOR I=0 TO 2047
30 A=VPEEK(BASE(2)+I)
40 VPOKE 4096+I,A
50 VPOKE 6144+I,A
60 NEXT
100 FOR I=65*8 TO 91*8-1
110 READ A:VPOKE 4096+I,A
120 NEXT
130 FOR I=97*8 TO 123*8-1
140 READ A:VPOKE 4096+I,A
150 NEXT
160 FOR I=48*8 TO 58*8-1
170 READ A:VPOKE 4096+I,A
180 NEXT
200 FOR I=65*8 TO 91*8-1
210 READ A:VPOKE 6144+I,A
220 NEXT
230 FOR I=97*8 TO 123*8-1
240 READ A:VPOKE 6144+I,A
250 NEXT
260 FOR I=48*8 TO 57*8-1
270 READ A:VPOKE 6144+I,A
280 NEXT
300 J=0:GOSUB 360
310 FOR J=8192 TO 16300 STEP 10
24
320 BASE(0)=J
330 GOSUB 360
340 NEXT J
350 GOTO 420
360 FOR I=0 TO 959
365 IF I>439 AND I<560 THEN VPO
KE BASE(0)+I,32:NEXT
370 VPOKE BASE(0)+I,194+J/1024
380 NEXT: A$="Esta Tela Se Inic
ia Em"+STR$(J)
390 FOR I=1 TO LEN(A$)
400 VPOKE BASE(0)+486+I,ASC(MID
$(A$,I,1))
410 NEXT:RETURN
420 BASE(0)=0:BASE(2)=2048
430 K$=INKEY$:IF K$="" THEN 430
440 IF K$=CHR$(31) AND BASE(0)>
8192 THEN BASE(0)=BASE(0)-1024:
GOTO 430
450 IF K$=CHR$(31) AND BASE(0)=
8192 THEN BASE(0)=0:GOTO 430
460 IF K$=CHR$(30) AND BASE(0)>
0 AND BASE(0)<15000 THEN BASE(0
)=BASE(0)+1024:GOTO 430
```

```
470 IF K$=CHR$(30) AND BASE(0)=
0 THEN BASE(0)=8192:GOTO 430
480 IF K$=CHR$(29) AND BASE(2)>
2048 THEN BASE(2)=BASE(2)-2048:
GOTO 430
490 IF K$=CHR$(28) AND BASE(2)<
6144 THEN BASE(2)=BASE(2)+2048:
GOTO 430
500 IF K$=CHR$(27) THEN BASE(0)
=0:BASE(2)=2048:END
510 GOTO 430
```

O objetivo desse programa é mostrar como usar toda a área da VRAM mesmo em **SCREEN 0**. Ao ser executado, o programa gasta um longo período lendo as linhas **DATA** do final da listagem. A seguir observamos que o vídeo é preenchido sucessivas vezes com caracteres diversos. Uma mensagem aparece no centro de cada tela, mas é apagada antes que possa ser lida. Isso acontece porque o programa está criando, na parte ociosa da VRAM, "telas secundárias" que, de certa forma, coexistem com as demais telas. Elas podem ser recuperadas logo após a parada do programa através das teclas do cursor.

O programa também já tem prontas três cópias da Tabela de Padrões, cada qual com um tipo de caractere, o que nos permite mudar o aspecto das letras recorrendo às teclas do cursor.

Para interromper o programa, use a tecla <**ESC**>. Se o programa for interrompido dessa maneira, ou porque ocorreu alguma mensagem de erro, e ele estiver em uma tela diferente da original, o cursor de texto não retornará à tela e o programa parecerá estar bloqueado. Para fazer o micro voltar ao normal, digite às cegas (nada aparecerá no vídeo) o comando abaixo: **SCREEN 0**

```
SCREEN 0
```

e pressione a tecla <**ENTER**>.

COMO FUNCIONA

Na linha 10, selecionamos o modo de textos e apagamos o rodapé da tela. O laço **FOR...NEXT** entre as linhas 20 e 60 cria duas cópias da Tabela de Padrões, que começa normalmente em 2048. As outras duas cópias terão endereços iniciais 4096 e 6144, respectivamente. Os laços seguintes obtêm nas linhas **DATA** os padrões que serão modificados. As cópias da tabela original são necessárias, porque apenas as letras e os números são redefinidos, excluindo-se, portanto, os demais símbolos.

As linhas 100 a 180 criam as letras futuristas na tabela que começa em 4096, enquanto as linhas 200 a 280 produzem as letras de fôrma, utilizando a tabela iniciada em 6144. Essas linhas se parecem muito com as primeiras do programa responsável pela mudança do conjunto de caracteres.

As telas secundárias são obtidas pelas linhas 300 a 420. A variável **J** determina o endereço inicial da tela produzida pela sub-rotina da linha 360. Assim, a primeira tela começando em 0 é criada pela linha 300.

A sub-rotina iniciada a partir da linha 360 preenche as diferentes telas com vários caracteres gráficos usando o comando **VPOKE** na linha 370. Como o código do caractere depende da variável **J**, ele resultará diferente a cada nova tela. Além disso, na linha 380 surge uma mensagem que identifica a tela através de seu endereço inicial. O laço das linhas 390 a 410, usando o comando **VPOKE**, imprime a mensagem. Esta se destaca por meio de um espaço claro criado pela linha 365.

A sub-rotina da linha 360 é chamada pela primeira vez na linha 300, para criar a tela original. A posição da tela pode ser mudada através da repetição da linha 320 para diferentes valores da variável **J**. Nessa linha, a variável **BASE(0)** e, portanto, o endereço inicial da Tabela de Nomes, assume o valor **J**. Para cada nova tela, a sub-rotina 360 é chamada outra vez pela linha 330. Depois de produzidas todas as telas secundárias, o programa prossegue na linha 420, onde os valores de **BASE(0)** e de **BASE(2)** são restituídos ao normal.

A porção do programa que abrange as linhas 430 a 510 torna possível a exibição instantânea de cada uma das telas criadas, escritas com qualquer dos três conjuntos de caracteres. Esta mudança é obtida pela atribuição de um novo valor a uma variável **BASE**. As linhas 440 e 450 permitem "avançar" dentro da VRAM através da tecla "seta para baixo". Novas telas vão sendo mostradas instantaneamente, à medida que se altera o valor de **BASE(0)**. Por outro lado, as linhas 460 e 470 possibilitam "retroceder" dentro da VRAM. São necessárias duas linhas para cada movimento, devido ao grande intervalo entre a tela original (endereço inicial 0) e as secundárias. Este espaço é ocupado pelas Tabelas de Padrões.

Com as linhas 480 e 490 podemos mudar instantaneamente o conjunto de caracteres, alterando o valor de **BASE(2)**, endereço inicial da Tabela de Padrões. Para isso, o programa detecta as teclas "seta para a direita" e "seta para a esquerda".

A linha 500 possibilita terminar o programa sem maiores problemas através da tecla <**ESC**>. Essa linha restitui **BASE(0)** e **BASE(2)** aos seus valores iniciais antes de encerrar.

Este programa poderia ser mais ilustrativo se preenchêssemos cada tela secundária com um texto, de modo que parecessem páginas de um livro. Trata-se, entretanto, de uma tarefa difícil em razão da enorme quantidade de linhas necessárias.

ALGUMAS LIMITAÇÕES

Interrompido o programa, ainda é possível obter uma mudança automática do conjunto de caracteres digitando:

```
BASE(2)=4096
```

ou

```
BASE(2)=6144
```

Para recuperar o conjunto original, digite o seguinte:

```
BASE(2)=2048
```

ou

```
SCREEN 0
```

Ao fazer isto, o usuário notará algo estranho no cursor: normalmente quando o sobrepomos a um caractere na tela, esse caractere se inverte. Por exemplo: se era claro sobre fundo escuro, torna-se escuro, sobre fundo claro. Quando mudamos a posição da Tabela de Padrões, entretanto, o cursor permanece como um espaço em branco, que apaga o caractere subjacente e desaparece na porção vazia da tela.

A inversão do caractere sob o cursor é realizada na Tabela de Padrões, pelo sistema operacional do MSX. Isto se dá através das posições 2040 a 2047 da VRAM. Quando mudamos a **BASE(2)**, o processo não é mais visível.

Comandos como **PRINT**, **TAB** e **LOCATE** só funcionam quando a **BASE(0)** tem valor 0, ou seja, só imprimem na Tabela de Nomes em sua posição original. Portanto, apenas por intermédio de **VPOKE** é possível imprimir nas telas secundárias; todas as mensagens de erro são impressas na Tabela de Nomes original, mesmo que uma tela secundária esteja sendo mostrada no vídeo.

Além de tudo isso, existem algumas limitações nos valores da instrução **BASE**. O endereço inicial de uma tabela de nomes — **BASE(0)** no caso de **SCREEN 0** — deverá ser múltiplo de 1024. Da mesma forma, o endereço inicial de uma tabela de padrões — **BASE(2)** em **SCREEN 0** — terá que ser múltiplo de 2048. Voltaremos a este assunto oportunamente, quando explicarmos os registros do chip de vídeo-VDP.

DESENHO ARQUITETÔNICO (1)

- O MÉTODO TRADICIONAL
- COMO USAR PAPEL E CANETA
- TIRE AS MEDIDAS
- DESENHOS E ESCALA
- O GIRO DAS PEÇAS

Deixe para o micro o trabalho "pesado" do planejamento de um ambiente. Com este projeto computadorizado, você pode distribuir seus móveis com o simples toque de algumas teclas.

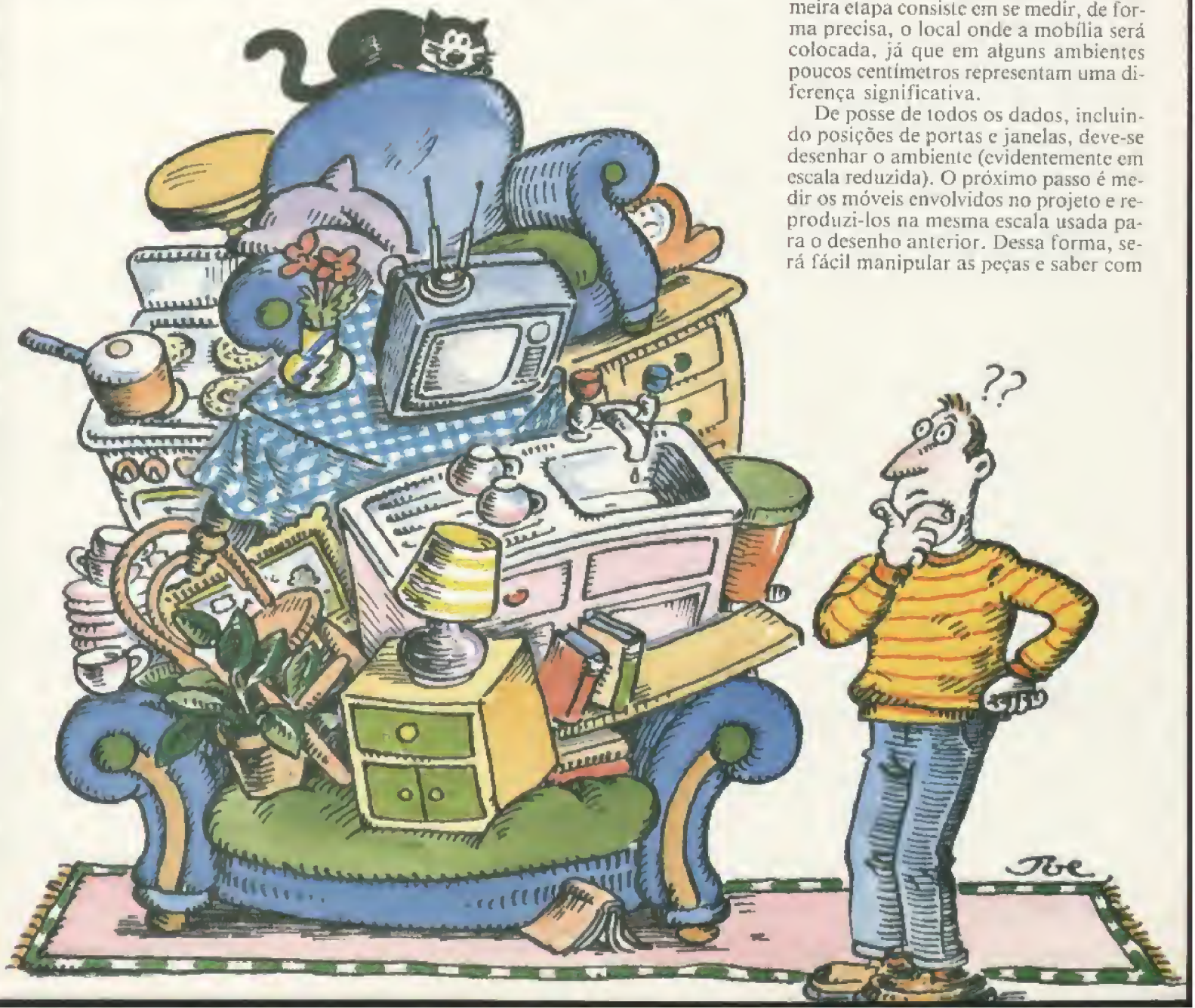
Conseguir uma melhor disposição dos móveis num ambiente nem sempre é algo fácil, e certamente é uma tarefa cansativa. Normalmente, optamos por arrastar a mobília de um lado para outro, até encontrarmos a melhor posição para as diferentes peças. O desânimo aparece quando constatamos que nem tudo se encaixa nos espaços disponíveis.

Um método mais simples consiste em fazer um planejamento prévio, desenhando cada móvel, com precisão, para em seguida adequá-los ao ambiente (obviamente é mais simples movimentar um retângulo que arrastar um sofá).

Apresentamos aqui, entretanto, uma terceira alternativa ainda mais fácil e moderna, utilizando seu microcomputador pessoal. O vídeo gráfico pode substituir o trabalho do papel e da caneta, com a grande vantagem de permitir fáceis correções. Além disso, os detalhes da mobília podem ser armazenados na memória do computador, eliminando a necessidade de redesenhar uma peça cada vez que quisermos movê-la.

O projeto computadorizado é semelhante ao projeto feito em papel. A primeira etapa consiste em se medir, de forma precisa, o local onde a mobília será colocada, já que em alguns ambientes poucos centímetros representam uma diferença significativa.

De posse de todos os dados, incluindo posições de portas e janelas, deve-se desenhar o ambiente (evidentemente em escala reduzida). O próximo passo é medir os móveis envolvidos no projeto e reproduzi-los na mesma escala usada para o desenho anterior. Dessa forma, será fácil manipular as peças e saber com



exatidão, por exemplo, qual a melhor posição para o piano na sala ou se a geladeira caberá em um determinado canto da cozinha.

Apresentamos aqui apenas metade do programa, que é composto de sete opções. O restante virá no próximo artigo.

As sete opções oferecidas no menu principal são as seguintes:

Opção 1 - Reproduz o ambiente: tomando por base sua maior medida, desenha-o em escala que caiba na tela do micro. As dimensões devem ser indicadas em metros, junto com suas respectivas posições (**C**ima, **B**aixo, **E**squerda, **D**ireita). Na representação das paredes, consideraremos duas distâncias e duas direções, permitindo, assim, o desenho na diagonal. Na opção 1, especificamos também a posição das portas e janelas.

Opção 2 - Possibilita a movimentação das peças da mobília já definidas no programa. São móveis de cozinha — talvez o ambiente mais difícil de ser planejado — e incluem armário, fogão, máquina de lavar louça, pia e geladeira, todos com medidas padronizadas. Qualquer alteração desejada deverá ser feita através da opção 3. O mesmo desenho de uma peça poderá ser colocado em diferentes lugares. Dessa forma, é possível, por exemplo, dispor quatro cadeiras iguais ao redor de uma mesa, posicionando a mesma definição da peça nos lugares escolhidos. Tudo que temos a fazer é selecionar o móvel acionando algumas teclas.

Opção 3 - Permite redefinir as peças já contidas no programa e incluir, no máximo, outras cinco, desde que não tenham mais que dez lados — o que é suficiente para desenhar qualquer móvel. As novas peças serão desenhadas automaticamente, tomando como base a escala da reprodução do ambiente.

Opção 4 - Permite salvar nosso projeto (opção um) e seu conteúdo (opções 2 e 3) em disco ou fita.

Opção 5 - Carrega um projeto e seu conteúdo de uma fita ou disco.

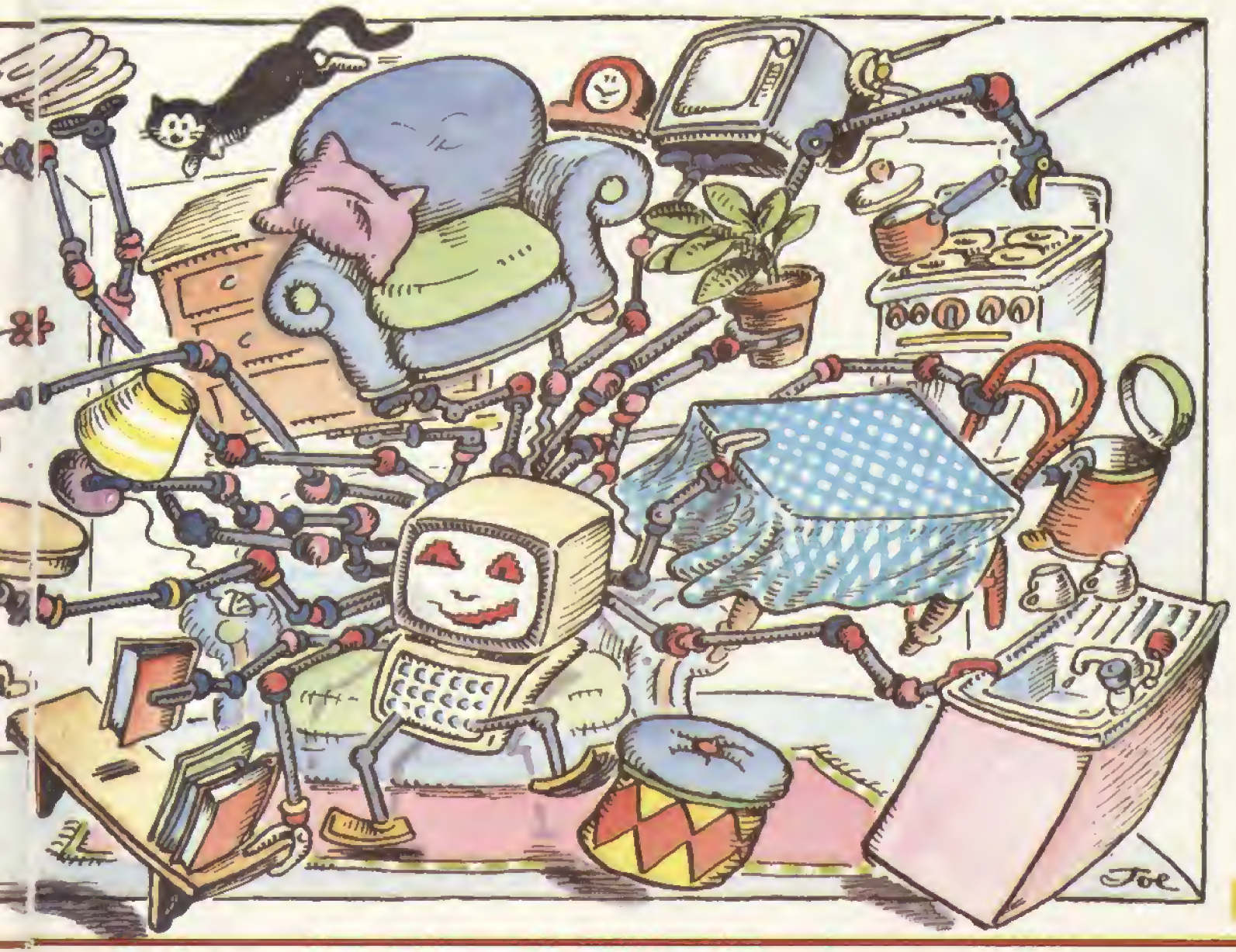
Opção 6 - Cuida da impressão. Como apenas o Spectrum é capaz de imprimir a cópia do projeto diretamente, esta opção ficará incompleta para os demais micros. Ela pode ser usada para chamar uma rotina que despeje o conteúdo da tela na impressora ou um outro programa em BASIC que faça a mesma tarefa. Em um próximo artigo, aprenderemos como criar essa rotina e obter cópias impressas de nossos projetos.

Opção 7 - É incumbida de terminar e sair do programa.

A segunda parte deste programa trará uma explicação mais detalhada de cada opção, além de conter a outra metade da listagem do programa. Mas atenção: é necessário digitar as duas partes para que o programa rode com segurança; portanto, grave a primeira antes de digitar a segunda.

```
10 PCLEAR 8:CLEAR 2000
20 DEF FNA(XM)=1.9*SC*XM
30 DIM O$(9),S(10)
40 O$(0)="DD100;DC50;DE100;DB50
;BD8;BC4;DD40;DC34;DE40;DB34;"
50 O$(1)="DD50;DC60;DE50;DB60;B
D10;BC10;DD10;DC10;DE10;DB10;BD
20;DD10;DC10;DE10;DB10;BC20;DC1
0;DD10;DB10;DE10;BE10;DE10;DC10
;DD10;DB10;BE20;BC15;DD50;"
60 O$(2)="DD100;DC60;DE100;DB60
;"
70 O$(3)="DD30;DC30;DE30;DB30;D
C20;DD30;"
80 O$(4)="DD60;DC60;DE60;DB60;"
90 CLS
100 PRINT @96,TAB(6)"1: PLANEJA
R AMBIENTE"
110 PRINT TAB(6)"2: DESENHAR LA
YOUT"
120 PRINT TAB(6)"3: DESENHAR MO
BILIA"
130 PRINT TAB(6)"4: GRAVAR PROJ
ETO"
140 PRINT TAB(6)"5: CARREGAR PR
OJETO"
150 PRINT TAB(6)"6: IMPRIMIR PR
OJETO"
160 PRINT TAB(6)"7: SAIDA"
170 PRINT @422,"FACA A OPCAO";:
INPUT N
180 IF N<1 OR N>7 THEN 90
190 IF N=2 AND F1=0 THEN CLS:PR
INT"VOCE DEVE PRIMEIRO SELECION
AR   OPCAO 1":SOUND 1,20:GOTO 9
0
200 IF N=1 THEN F1=1
210 ON N GOTO 350,1120,830,1700
,1750,1790,230
220 GOTO 90
230 CLS:PCLS:END
240 RF=0:COLOR 0
250 LINE(200,0)-(255,191),PSET,
B
260 DRAW"BM206,10;S4;A0;NR4D6R4
BR2U6D3R4D3U6BR2D6R4U6NL4BR2D6R
4U6NL4BR2NR4D3R4D3NL4BR2NR4U3NR
2U3R4BR4BD2DBD2D"
270 IF RF=1 THEN RETURN
280 DRAW"BM203,30;D6R2NU3R2U6BR
6R4D6L4U6BR10D6R4U3L4R3U3L3BR10
NR4D3R4D3NL4BR2U6R4D3L4BR6U3NR4
D6R4"
290 DRAW"BM218,43;D6U3NR2U3R4BF
6NR4D6R4"
300 RETURN
310 RF=1:COLOR 0,1:LINE(201,1)-
(254,190),PRESET,BF:GOSUB 250
320 DRAW"BM208,30;ND6R4D3L4BR10
BU3D6R3E1U4H1L3BR10;ND6R4D3L4R1
F3BR6U6R4L4D3R2"
330 RETURN
340 DRAW"BM213,70;D4F1R2E1U4H1L
2G1BU1BR8BD3R3BR5U3R4D3NL4D3BR8
UBU2U2R2U2L4D2":RETURN
350 CLS
360 PRINT"COMPRIMENTO MAXIMO DO
 COMODO(M)"
370 INPUT LE
380 IF LE>100 OR LE<3 THEN 350
390 SC=100/LE
400 PMODE 4,1:COLOR 0,1:PCLS:SC
REEN 1,0
410 XM=0:YM=LE
420 GOSUB 240
430 XX=FNA(XM):YY=FNA(YM):IF PP
OINT(XX,YY)=1 THEN PSET(XX,YY,0
) ELSE PSET(XX,YY,1)
440 I$=INKEY$:IF I$="" THEN 430
450 OX=XM:OY=YM
460 IF I$=" " THEN COLOR 0:GOTO
 530
470 IF I$="B" THEN COLOR 1:GOTO
 530
480 IF I$="C" THEN 400
490 IF I$="F" THEN FOR K=1 TO 4
```

```
:PCOPY K TO K+4:NEXT:GOTO 90
500 IF I$="W" THEN 710
510 IF I$="O" THEN 800
520 GOTO 430
530 CLS
540 FOR A=1 TO 2
550 PRINT"DIRECAO";A;" C/B/E/D
";:INPUT D$(A)
560 IF D$(A)="" THEN 600
570 PRINT "DISTANCIA";A;:INPUT
D(A)
580 IF INSTR(1,"CBED",D$(A))=0
THEN 550
590 NEXT
600 SCREEN 1,0:FOR A=1 TO 2
610 IF D$(A)="" THEN 670
620 IF D$(A)="E" THEN XM=XM-D(A
)
630 IF D$(A)="D" THEN XM=XM+D(A
)
640 IF D$(A)="C" THEN YM=YM-D(A
)
650 IF D$(A)="B" THEN YM=YM+D(A
)
660 NEXT A
670 IF XM<0 OR XM>LE OR YM<0 OR
 Y>LE THEN SOUND 1,2:XM=OX:YM=O
Y:GOTO 430
680 X1=FNA(OX):Y1=FNA(OY):X2=FN
A(XM):Y2=FNA(YM)
690 LINE(X1,Y1)-(X2,Y2),PSET
700 GOTO 430
710 CLS:INPUT"DIRECAO C/B/E/D";
D$
720 INPUT "DISTANCIA";D
730 X1=FNA(OX):Y1=FNA(OY)
740 POKE 178,2
750 IF D$="E" THEN XM=XM-D:X2=F
NA(XM):LINE(X1,Y1)-(X2,Y1+3),PS
ET,BF
760 IF D$="D" THEN XM=XM+D:X2=F
NA(XM):LINE(X1,Y1)-(X2,Y1+3),PS
ET,BF
770 IF D$="C" THEN YM=YM-D:Y2=F
NA(YM):LINE(X1,Y1)-(X1+3,Y1),PS
ET,BF
780 IF D$="B" THEN YM=YM+D:Y2=F
NA(YM):LINE(X1,Y1)-(X1+3,Y2),PS
ET,BF
790 SCREEN 1,0:GOTO 430
800 CLS:INPUT"DIRECAO C/B/E/D";
D$(1)
810 INPUT"DISTANCIA";D(1)
820 D$(2)="":COLOR 1:GOTO 600
830 CLS
840 INPUT"DEFINIR ITEM NUMERO
(0-9)";N
850 IF N<0 OR N>9 THEN 830
860 O$(N)=""
870 PRINT"USE <ESPACO> PARA LIN
HA":PRINT"USE 'B' PARA LINHA VA
ZIA":FOR D=1 TO 1000:NEXT
880 PMODE 4,1:COLOR 0,1:PCLS:SC
REEN 1,0
890 X=75:Y=145
900 DRAW"BM75,150;R100NG3NH3;BM
70,145;U100NF3NG3"
910 IF PPOINT(X,Y)=0 THEN PSET(
X,Y,1) ELSE PSET (X,Y,0)
920 I$=INKEY$:IF I$="" THEN 910
930 OX=X:OY=Y
940 IF I$="C" THEN 830
950 IF I$="F" THEN 90
960 IF I$=" " THEN COLOR 0
970 IF I$<>" " THENCOLOR 1
```

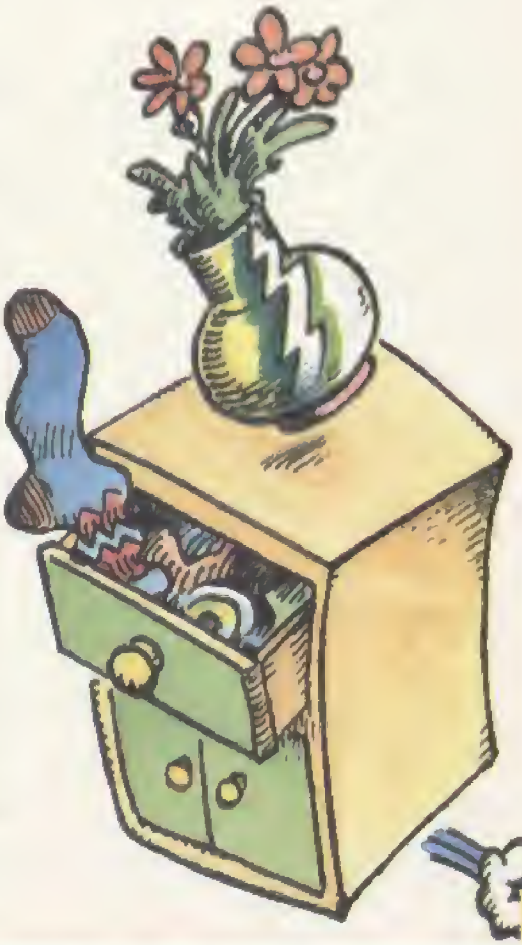


```
980 IF I$=" " THEN O$(N)=O$(N)+
"D" ELSE O$(N)=O$(N)+"B"
990 CLS:INPUT "DIRECAO C/B/E/D"
;D$
1000 IF INSTR(1,"CBED",D$)=0 TH
EN 990
1010 INPUT"DISTANCIA (CM)";D
1020 IF D<=0 OR D>200 THEN 1010
1030 SCREEN 1,0
1040 IF D$="E" THEN X=X-D/2
1050 IF D$="D" THEN X=X+D/2
1060 IF D$="C" THEN Y=Y-D/2
1070 IF D$="B" THEN Y=Y+D/2
1080 IF X<75 OR X>175 OR Y<45 O
R Y>145 THEN SOUND 1,3:X=OX:Y=O
Y:GOTO 910
1090 LINE(OX,OY)-(X,Y),PSET
1100 O$(N)=O$(N)+D$+STR$(D)+";"
1110 GOTO 910
1120 FOR K=1 TO 4:PCOPY K+4 TO
K:NEXT:CLS
1130 PMODE 4,1:SCREEN 1,0
1140 GOSUB 310
1150 X=128:Y=96:RT=0
1160 IF X<3 THEN X=3
1170 IF Y<3 THEN Y=3
1180 GET(X-3,Y-3)-(X+3,Y+3),S,G
1190 DRAW"C0;BM"+STR$(X)+","+ST
R$(Y)+"NU3ND3NL3NR3"
1200 I$=INKEY$:IF I$="" THEN 12
00
```

```
  5 POKE 23658,8
 10 BORDER 0: PAPER 0: INK 4:
 CLS
 12 GOSUB 8000
 20 GOSUB 7000: PRINT  INVERSE
 1;AT 2,23;"[1]AMB";AT 3,23;"[
 2]PLAN";AT 4,23;"[3]MOB";AT 5
 ,23;"[4]GRAV";AT 6,23;"[5]CAR
 R";AT 7,23;"[6]IMPR";AT 8,23;
 "[7]SAIDA"
 30 LET K$="1234567": GOSUB
 7040: GOSUB 7000: IF Z=55
 THEN  STOP
 40 GOSUB 1000*(Z-49)+1000:
 GOTO 20
1000 LET NF=1: GOSUB 6080
1005 PLOT 0,0: LET X=0: LET Y=0
1010 PRINT  PAPER 2; INK 6;AT 2
1,22;"MAXIMO=";MAX
1015 PRINT  INVERSE 1;AT 2,22;"
[J]JANELA";AT 3,22;"[P]PORTA";A
T 4,22;"[B]BRANCO";AT 5,22;"[ ]
DESENHA";AT 6,22;"[S]SAIDA";AT
7,22;""
1016 LET K$="JPBS ": GOSUB 7040
: INK 3*(Z=74)+2*(Z=80)+7*(Z=32
)
1020 IF Z=83 THEN  INK 4: RETUR
N
1025 GOSUB 7010: LET DX=X+D*((D
$="D")*SC)-D*((D$="E")*SC): LET
 DY=Y+D*((D$="C")*SC)-D*((D$="B
")*SC): GOSUB 7010
1030 LET DX=DX+D*((D$="D")*SC)-
D*((D$="E")*SC): LET DY=DY+D*((
D$="C")*SC)-D*((D$="B")*SC)
1032 IF DX>175 OR DX<0 OR DY>17
5 OR DY<0 THEN  PRINT  FLASH 1;
AT 7,23;"ERRO": PAUSE 100: GOTO
 1015
1035 DRAW DX-X,DY-Y: LET X=PEEK
 23677: LET Y=PEEK 23678: LET X
$=STR$ (X/SC): LET Y$=STR$ (Y/S
C)
1037 IF LEN X$<3 THEN  LET X$=F
$( TO 3-LEN X$)+X$
1038 IF LEN Y$<3 THEN  LET Y$=F
$( TO 3-LEN Y$)+Y$
1039 LET X$=X$( TO 3): LET Y$=Y
$( TO 3): PRINT  INK 7;AT 10,24
;"DX=";X$;AT 11,24;"DY=";Y$
1040 GOTO 1016
2000 IF NF=0 THEN  RETURN
2005 FOR N=1 TO 10: LET Q$="["+
STR$ N+"]"+O$(N): PRINT  INVERS
E 1;AT N*2,24+4-LEN Q$;Q$: NEXT
 N
2010 LET R=0: GOSUB 6060: LET C
U=2
2020 PRINT  INVERSE 1;AT CU,29;
CHR$ 144: FOR N=1 TO 10: NEXT N
: PRINT  INVERSE 1;AT CU,29;" "
2030 LET K$=INKEY$: LET CU=CU-2
*(((K$="7")*(CU>2)))+2*(((K$="6
")*(CU<20)))
2040 IF K$<>"S" THEN  GOTO 2020
2050 LET OB=CU/2: LET OX=85: LE
T OY=85
2057 LET F=1: GOSUB 6010
2060 GOSUB 7000: PRINT AT 5,24;
 FLASH 1;"OBJ=";O$(OB); FLASH 0
;AT 7,22; INVERSE 1;"[5-8]MOVE"
;AT 8,22;"[C]COLOCA";AT 9,22;"[
H]HORARIO";AT 10,22;"[A]ANTHORA
";AT 11,22;"[S]SAIDA"
2070 LET K$="5678CAHS": GOSUB 7
040: IF Z=83 THEN  LET R=0: LET
 F=1: GOSUB 6010: GOTO 6060
2080 LET F=1: GOSUB 6010: LET O
X=OX+2*((Z=56)*(OX<175))-2*((Z=
53)*(OX>1)): LET OY=OY+2*((Z=55
)*(OY<175))-2*((Z=54)*(OY>0))
2085 IF Z=72 OR Z=65 THEN  GOSU
B 6020
2090 GOSUB 6010
2100 IF Z=67 THEN  LET F=0: GOS
UB 6010: GOSUB 7000: GOTO 2000
2110 GOTO 2070
3000 PRINT AT 2,24;"PROJETO"
3012 INPUT "ENTRE FIGURA A DEFI
NIR ";OB: IF OB<1 OR OB>10 THEN
  GOTO 3012
3013 INPUT "DIGITE NUMERO DE LA
DOS (1-15) ? ";S(OB): IF S(OB)<
1 OR S(OB)>15 THEN  GOTO 3013
3014 INPUT "CODIGO DE DUAS LETR
AS ?";O$(OB)
3016 LET R=0: GOSUB 6060: LET O
X=80: LET OY=80
3017 FOR S=1 TO S(OB)*2 STEP 2
3020 FOR N=1 TO 2: GOSUB 7010:
LET D=D/125: LET O(OB,S)=O(OB,S
)+((D*(-1*(D$="E")+(D$="D")))):
 LET O(OB,S+1)=O(OB,S+1)+((D*(-
1*(D$="B")+(D$="C"))))
3030 NEXT N
3040 NEXT S
3050 LET F=1: GOSUB 6010: INPUT
 "ESTA CORRETO (S/N) ?";S$: IF
S$="N" THEN  GOSUB 6010: FOR N=
1 TO S(OB)*2+1: LET O(OB,N)=0:
NEXT N: GOTO 3012
3060 GOSUB 6010: RETURN
4000 GOSUB 6200
4015 IF Z=83 THEN  SAVE E$SCREE
N$ : RETURN
4020 SAVE E$ DATA O()
4030 SAVE E$ DATA S()
4040 SAVE E$ DATA O$()
4050 RETURN
5000 GOSUB 6200
5010 IF Z=83 THEN  LOAD E$SCREE
N$ : RETURN
5020 LOAD E$ DATA O()
5030 LOAD E$ DATA S()
5040 LOAD E$ DATA O$()
5050 RETURN
6000 COPY : RETURN
6010 OVER F: INK 7: PLOT OX,OY:
 FOR N=1 TO (S(OB)*2) STEP 2: D
RAW O(OB,N)*CY-O(OB,N+1)*CX,O(O
B,N)*CX+O(OB,N+1)*CY: NEXT N
6014 OVER 0: INK 4: RETURN
6020 LET R=R+(2*(Z=65))-(2*(Z=7
2)): IF R>360 THEN  LET R=0
6030 IF R<0 THEN  LET R=360-R
6060 LET A=R*(PI/180): LET CY=S
C*COS A: LET CX=SC*SIN A: RETUR
N
6080 INPUT "ENTRE MAXIMA DIMENS
AO ?";MAX
6090 LET SC=175/MAX
6100 RETURN
6200 PRINT  INVERSE 1;AT 10,23;
"[T]TELA";AT 11,23;"[D]DADOS":
LET K$="TD": GOSUB 7040: INPUT
"ENTRE NOME DO ARQUIVO";E$: RET
URN
7000 FOR N=0 TO 21: PRINT  PAPE
R 4;AT N,22;"          ": NEXT
N: PRINT AT 0,25;"MENU": RETURN
```

```
5 SCREEN 2,0:COLOR 1,15,15
10 KEY OFF:GOTO 2000
20 DEF FNA(XM)=1.9*SC*XM
30 DIM O$(9),S(10)
40 O$(0)="DD100;DC50;DE100;DB50
;BD8;BC8;DD40;DC34;DE40;DB34;"
50 O$(1)="DD50;DC60;DE50;DB60;B
D10;BC10;DD10;DC10;DE10;DB10;BD
20;DD10;DC10;DE10;DB10;BC20;DC1
0;DD10;DB10;DE10;BE10;DE10;DC10
;DD10;DB10;BE20;BC15;DD50;"
60 O$(2)="DD100;DC60;DE100;DB60
;"
70 O$(3)="DD30;DC30;DE30;DB30;D
C20;DD30;"
80 O$(4)="DD60;DC60;DE60;DB60;"
85 CLS
90 SCREEN 0
100 LOCATE 8,5:PRINT"1: PLANEJA
R AMBIENTE"
110 LOCATE 8,6:PRINT"2: DESENHA
R LAYOUT"
120 LOCATE 8,7:PRINT"3: DESENHA
R MOBILIA"
130 LOCATE 8,8:PRINT"4: GRAVAR
  PROJETO"
140 LOCATE 8,9:PRINT"5: CARREGA
R PROJETO"
150 LOCATE 8,10:PRINT"6: IMPRIM
IR PROJETO"
160 LOCATE 8,11:PRINT"7: TERMIN
```

```
AR"
170 LOCATE 8,14:INPUT"SUA ESCOL
HA ";N
180 IF N<1 OR N>7 THEN 170
190 IF N=2 AND F1=0 THEN LOCATE
 5,18:PRINT"<escolha primeiro a
 opção 1>":GOTO 170
200 IF N=1 THEN F1=1
210 ON N GOTO 350,1120,830,1700
,1750,1790,230
220 GOTO 90
230 CLS:END
240 RF=0
250 LINE(200,0)-(255,191),1,B
260 DRAW"BM206,10;S4;A0;NR4D6R4
BR2U6D3R4D3U6BR2D6R4U6NL4BR2D6R
4U6NL4BR2NR4D3R4D3NL4BR2NR4U3NR
2U3R4BR4BD2DBD2D"
270 IF RF=1 THEN RETURN
280 DRAW"BM203,30;D6R2NU3R2U6BR
6R4D6L4U6BR10D6R4U3L4R3U3L3BR10
NR4D3R4D3NL4BR2U6R4D3L4BR6U3NR4
D6R4"
290 DRAW"BM218,43;D6U3NR2U3R4BR
6NR4D6R4"
300 RETURN
310 RF=1:LINE(201,1)-(254,190),
15,BF:GOSUB 250
320 DRAW"BM208,30;ND6R4D3L4BR10
BU3D6R3E1U4H1L3BR10;ND6R4D3L4R1
F3BR6U6R4L4D3R2"
330 RETURN
340 DRAW"BM213,70;D4F1R2E1U4H1L
2G1BU1BR8BD3R3BR5U3R4D3NL4D3BR8
UBU2U2R2U2L4D2":RETURN
350 CLS
360 PRINT"COMPRIMENTO MAXIMO DO
 AMBIENTE"
370 INPUT"(em metros)";LE
380 IF LE<3 OR LE>100 THEN 350
390 SC=100/LE
400 SCREEN 2
410 XM=0:YM=LE
420 GOSUB 240:A=USR(0)
430 XX=FNA(XM):YY=FNA(YM):IF PO
INT(XX,YY)<>1 THEN PSET(XX,YY),
1
440 I$=INKEY$:IF I$="" THEN 430
450 OX=XM:OY=YM
455 D$(1)="":D$(2)=""
460 IF I$=" " THEN CL=1:GOTO 530
470 IF I$="B" THEN CL=15:GOTO 5
30
480 IF I$="C" THEN 400
490 IF I$="F" THEN A=USR(0):GOT
090
500 IF I$="W" THEN 710
510 IF I$="O" THEN CL=15:GOTO 8
00
520 GOTO 430
530 SCREEN 0
540 FOR A=1 TO 2
550 PRINT"Direção";A;" C/B/D/E"
;:INPUT D$(A)
560 IF D$(A)="" THEN 600
570 PRINT"Distância";A;" (metro
s)";:INPUT D(A)
580 IF INSTR(1,"CBDE",D$(A))=0
THEN550
590 NEXT
600 SCREEN 2:A=USR2(0):A=USR1(0
)
605 FOR A=1 TO 2
610 IF D$(A)="" THEN 670
620 IF D$(A)="E" THEN XM=XM-D(A
)
630 IF D$(A)="D" THEN XM=XM+D(A
)
640 IF D$(A)="C" THEN YM=YM-D(A
)
650 IF D$(A)="B" THEN YM=YM+D(A
)
660 NEXT A
670 IF XM<0 OR XM>LE OR YM<0 OR
 YM>LE THEN BEEP:XM=OX:YM=OY:GO
TO 430
680 X1=FNA(OX):Y1=FNA(OY):X2=FN
A(XM):Y2=FNA(YM)
690 LINE(X1,Y1)-(X2,Y2),CL
695 A=USR(0)
700 GOTO 430
710 SCREEN 0:INPUT"Direção   C/
B/D/E";D$
720 INPUT"Distância ";D
730 X1=FNA(OX):Y1=FNA(OY)
735 SCREEN2:A=USR2(0):A=USR1(0)
750 IF D$="E" THEN XM=XM-D:X2=F
NA(XM):LINE(X1,Y1+1)-(X2,Y1+1),
1:LINE(X1,Y1)-(X2,Y2),1
760 IF D$="D" THEN XM=XM+D:X2=F
NA(XM):LINE(X1,Y1+1)-(X2,Y1+1),
1:LINE(X1,Y1)-(X2,Y2),1
770 IF D$="C" THEN YM=YM-D:Y2=F
NA(YM):LINE(X1+1,Y1)-(X1+1,Y2),
1:LINE(X1,Y1)-(X2,Y2),1
780 IF D$="B" THEN YM=YM+D:Y2=F
NA(YM):LINE(X1+1,Y1)-(X1+1,Y2),
1:LINE(X1,Y1)-(X2,Y2),1
790 A=USR(0):GOTO 430
800 CLS:INPUT"Direção   C/B/D/E
";D$(1)
810 INPUT"Distância ";D(1)
820 D$(2)="":GOTO 600
830 CLS
840 INPUT"DEFINIR A PEÇA NUMERO
 (0-9)";N
850 IF N<0 OR N>9 THEN 830
860 O$(N)=""
870 SCREEN 2:RF=1:GOSUB 250
880 DRAW"BM218,43;NR4D3R4D3NL4B
R2U6R4D3L4BR6U3NR4D6R4;BM210,56
;D6R4U3L4R3U3L3BR15D6U3NR2U3R4B
R10NR4D6R4"
890 X=75:Y=145
900 DRAW"BM75,150;R100NG3NR3;BM
70,145;U100NF3NG3"
910 IF POINT(X,Y)<>1 THEN PSET(
X,Y),1
920 I$=INKEY$:IF I$="" THEN 920
925 A=USR3(0)
930 OX=X:OY=Y
940 IF I$="C" THEN 830
950 IF I$="F" THEN 90
960 IF I$=" " THEN CL=1
970 IF I$<>" " THEN CL=15
980 IF I$=" " THEN O$(N)=O$(N)+
"D" ELSE O$(N)=O$(N)+"B"
990 SCREEN0:INPUT"Direção   C/B
/D/E";D$
1000 IF INSTR(1,"CBDE",D$)=0 TH
EN 990
1010 INPUT"Distância (cms)";D
1020 IF D<=0 OR D>200 THEN 1010
1030 SCREEN2:A=USR2(0):A=USR4(0
)
1040 IF D$="E" THEN X=X-D/2
1050 IF D$="D" THEN X=X+D/2
1060 IF D$="C" THEN Y=Y-D/2
1070 IF D$="B" THEN Y=Y+D/2
1080 IF X<75 OR X>175 OR Y<45 O
R Y>145 THEN BEEP:X=OX:Y=OY:GOT
O 910
1090 LINE(OX,OY)-(X,Y),CL
1100 O$(N)=O$(N)+D$+MID$(STR$(D
),2)+";"
1110 GOTO 910
1120 SCREEN 2
1130 A=USR2(0):A=USR1(0)
1140 GOSUB 310
1150 X=128:Y=96:RT=0
1160 IF X<3 THEN X=3
1170 IF Y<3 THEN Y=3
1190 PUTSPRITE0,(X,Y),1,0
1200 I$=INKEY$:IFI$="" THEN 120
0
```



A MATEMÁTICA DA IRREGULARIDADE (2)

Neste artigo, você verá como aplicar o que aprendeu sobre dimensões fracionadas na simulação de imagens da natureza, como uma montanha ou um floco de neve.

No artigo da página 1356, vimos como obter imagens fascinantes por meio de programas recursivos muito simples. Essas imagens, também denominadas *figuras fractais*, são geradas matematicamente na fronteira entre o que chamamos de regular e irregular.

Com a técnica fractal podemos conseguir formas muito mais próximas das observadas na natureza que com os modelos construídos a partir da ciência tradicional. Mostraremos aqui como fazer simulações desse tipo no seu micro.

O primeiro programa desenha uma das formas mais simétricas da natureza: a forma hexagonal de um floco de neve.

```
  10 BORDER 0: PAPER 0: INK 5:
  BRIGHT 1: CLS
  20 LET AN=2*ATN (1)/3: LET S2
  =2/SQR (3)
  30 LET XC=127: LET YC=90: LET
  S=120: LET C=2
  50 GOSUB 1000
  60 STOP
1000 LET S=S/3: IF S<1 THEN  LE
T S=S*3: RETURN
1020 PLOT  INVERSE 1; OVER 1;IN
T (XC+S2*S*SIN (-AN)),(YC-S2*S*
COS (-AN)): FOR K=0 TO 8*ATN (1
)-AN STEP 2*AN
1030 DRAW XC+2*S*SIN (K)-PEEK 2
3677,YC-2*S*COS (K)-PEEK 23678
1040 DRAW XC+S2*S*SIN (K+AN)-PE
EK 23677,YC-S2*S*COS (K+AN)-PEE
K 23678
1050 NEXT K
1060 LET C=C-1: GOSUB 1000
1070 LET YC=YC-1.36*S: GOSUB 10
00
1080 LET YC=YC+.68*S: LET XC=XC
+1.19*S: GOSUB 1000
1090 LET YC=YC+1.36*S: GOSUB 10
00
1100 LET YC=YC+.68*S: LET XC=XC
-1.19*S: GOSUB 1000
1110 LET YC=YC-.68*S: LET XC=XC
-1.19*S: GOSUB 1000
1120 LET YC=YC-1.36*S: GOSUB 10
00
1130 LET YC=YC+.68*S: LET XC=XC
+1.19*S: LET S=S*3: LET C=C+1:
RETURN
```

```
10 PMODE 3,1:PCLS:SCREEN 1,0
20 AN=2*ATN(1)/3:S2=2/SQR(3)
30 XC=127:YC=95:S=135:C=4
50 GOSUB 1000
60 GOTO 60
1000 S=S/3:IF S<1 THEN S=S*3:RE
TURN
1010 IF C=2 THEN COLOR 4 ELSE I
F C=1 THEN COLOR 2 ELSE COLOR C
1020 DRAW"BM"+STR$(INT(XC+S2*S*
SIN(-AN)))+","+STR$(INT(YC-S2*S
*COS(-AN))):FOR K=0 TO 8*ATN(1)
-AN STEP 2*AN
1030 LINE -(XC+2*S*SIN(K),YC-2*
S*COS(K)),PSET
1040 LINE -(XC+S2*S*SIN(K+AN),Y
C-S2*S*COS(K+AN)),PSET
1050 NEXT:PAINT(XC,YC)
1060 C=C-1:GOSUB 1000
1070 YC=YC-1.36*S:GOSUB 1000
1080 YC=YC+.68*S:XC=XC+1.19*S:G
OSUB 1000
1090 YC=YC+1.36*S:GOSUB 1000
1100 YC=YC+.68*S:XC=XC-1.19*S:G
OSUB 1000
1110 YC=YC-.68*S:XC=XC-1.19*S:G
OSUB 1000
1120 YC=YC-1.36*S:GOSUB 1000
1130 YC=YC+.68*S:XC=XC+1.19*S:S
=S*3:C=C+1:RETURN
```

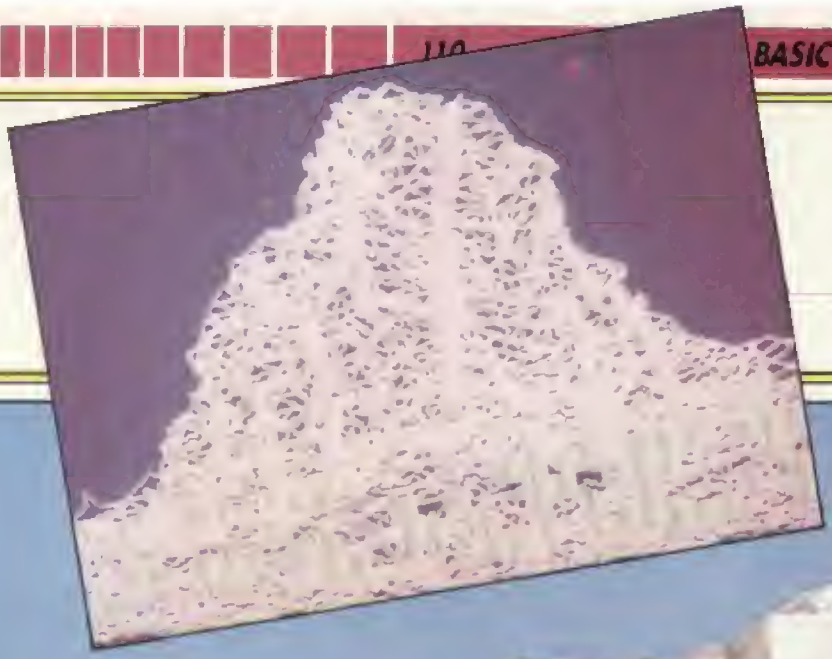
```
10  HGR2
20 AN = 2 *  ATN (1) / 3:S2 = 2
 /  SQR (3)
30 XC = 127:YC = 95:S = 135:C =
 4
50  GOSUB 1000
60  GOTO 60
1000 S = S / 3: IF S < 1 THEN S
 = S * 3: RETURN
1020  HPLOT  INT (XC + S2 * S *
  SIN ( - AN)), INT (YC - S2 *
S *  COS ( - AN)): FOR K = 0 TO
 8 *  ATN (1) - AN STEP 2 * AN
1030  HPLOT  TO XC + 2 * S *  S
IN (K),YC - 2 * S *  COS (K)
1040  HPLOT  TO XC + S2 * S *
SIN (K + AN),YC - S2 * S *  COS
  (K + AN)
1050  NEXT
1060 C = C - 1: GOSUB 1000
1070 YC = YC - 1.36 * S: GOSUB
1000
1080 YC = YC + .68 * S:XC = XC
+ 1.19 * S: GOSUB 1000
1090 YC = YC + 1.36 * S: GOSUB
1000
1100 YC = YC + .68 * S:XC = XC
- 1.19 * S: GOSUB 1000
1110 YC = YC - .68 * S:XC = XC
- 1.19 * S: GOSUB 1000
1120 YC = YC - 1.36 * S: GOSUB
1000
1130 YC = YC + .68 * S:XC = XC
+ 1.19 * S:S = S * 3:C = C + 1:
 RETURN
```

```
10 SCREEN 2
20 AN=2*ATN(1)/3:S2=2/SQR(3)
30 XC=127:YC=95:S=135:C=4
50 GOSUB 1000
60 GOTO 60
1000 S=S/3:IF S<1 THEN S=S*3:RE
TURN
1010 IF C=2 THEN COLOR 8 ELSE I
F C=1 THEN COLOR 11 ELSE COLOR
C
1020 DRAW "BM"+STR$(INT(XC+S2*S
*SIN(-AN)))+","+STR$(INT(YC-S2*
S*COS(-AN))):FOR K=0 TO 8*ATN(1
)-AN STEP 2*AN
1030 LINE-(XC+2*S*SIN(K),YC-2*S
*COS(K))
1040 LINE-(XC+S2*S*SIN(K+AN),YC
-S2*S*COS(K+AN))
1050 NEXT:PAINT(XC,YC)
1060 C=C-1:GOSUB 1000
1070 YC=YC-1.36*S:GOSUB 1000
1080 YC=YC+.68*S:XC=XC+1.19*S:G
OSUB 1000
1090 YC=YC+1.36*S:GOSUB 1000
1100 YC=YC+.68*S:XC=XC-1.19*S:G
OSUB 1000
1110 YC=YC-.68*S:XC=XC-1.19*S:G
OSUB 1000
1120 YC=YC-1.36*S:GOSUB 1000
1130 YC=YC+.68*S:XC=XC+1.19*S:S
=S*3:C=C+1:RETURN
```

Esse programa é baseado na *curva floco de neve*, originalmente desenhada por Von Koch. A figura criada se assemelha a um cristal de gelo em um floco de neve ou a uma ilha com litoral muito recortado. A linha 20 especifica um triângulo eqüilátero (seus ângulos medem 60 graus e os lados são iguais) com um fator de escala definido em S2. A linha 30 define as primeiras coordenadas X e Y do centro, um fator de escala inicial e a cor do primeiro desenho (exceto no Apple). A linha 50 chama a sub-rotina que desenha a estrela de seis pontas que compõe a figura.

SIMETRIA E ASSIMETRIA

Apesar da irregularidade do desenho, a simetria está presente na forma da estrela. Como ocorre com freqüência na



- MODELOS DE SIMETRIA
- PROGRAMA DO FLOCO DE NEVE
- FORMAS ASSIMÉTRICAS
- UMA MONTANHA NO MICRO
- GERADOR DE FIGURAS

natureza, a ordem e o caos se combinam nessa figura. Porém, muitas estruturas simuladas pela técnica fractal são totalmente assimétricas. É o caso das curvas de um rio, das crateras da Lua, das veias e artérias do corpo humano e do contorno das montanhas. O que distingue essas estruturas das formas simétricas geradas matematicamente são os elementos aleatórios que entram em sua composição. Podemos simular tais elementos no computador por meio da função **RND**. O programa que se segue gera a imagem de uma montanha:

```
10 BORDER 0: PAPER 0: INK 7:
BRIGHT 1: CLS
```

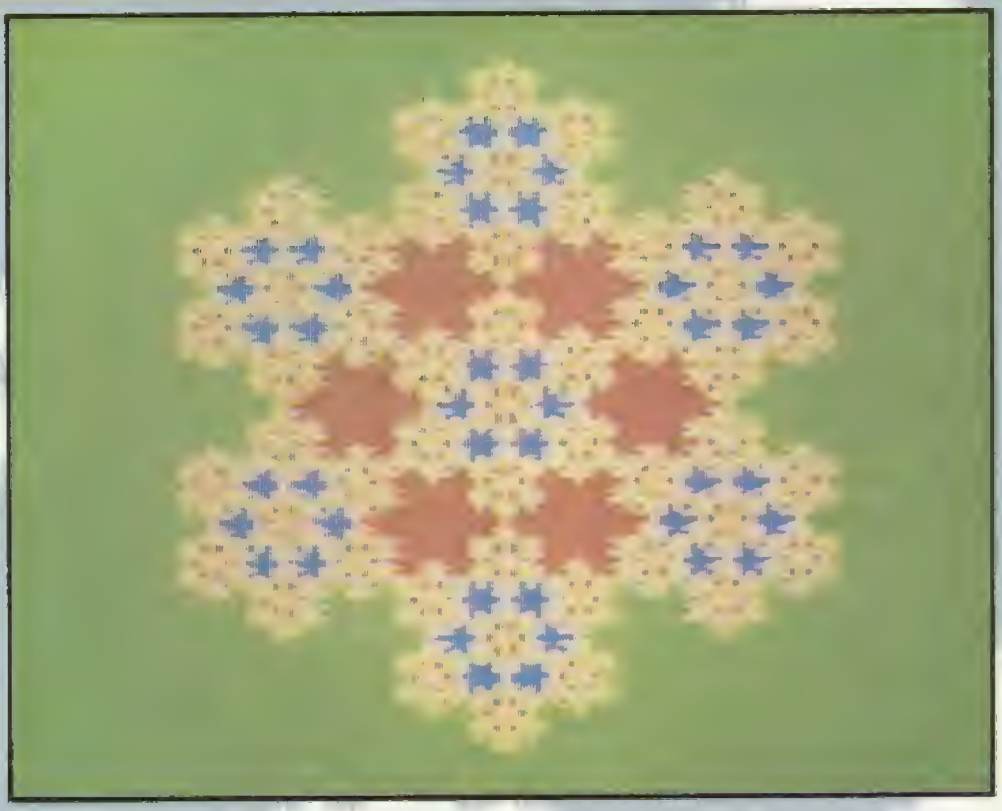
A técnica dos fractais é responsável pelas mais perfeitas imagens de formas naturais simuladas no computador. A construção de uma imagem complexa como a escarpa nevada desta página só é possível em computadores gráficos de maior porte. Porém, a mesma técnica pode ser utilizada em micros domésticos, com resultado semelhante ao mostrado na ilustração do alto.

```
15 PRINT AT 6,2; INVERSE 1;"
GERADOR DE MONTANHA FRACTAL "
20 DIM C(200,2,2): LET F=1:
LET G=2: LET C(1,1,2)=25: LET
C(1,1,1)=0
22 INPUT "DIGITE 'RESOLUCAO'
DA MONTANHA  [16-100] ? ";S
23 IF S<16 OR S>100 THEN
GOTO 22
24 INPUT "DIGITE GRAU DE 'RUG
OSIDADE'[1-5]?";RG
25 IF RG<1 OR RG>5 THEN  GOTO
 24
26 DEF FN R(X)=RG-((RND*X)*(2
*RG))
27 PAPER 1: CLS
30 LET L=230/S: LET H=L/(SQR
3)
40 FOR K=1 TO S+1: LET C(K,1,
1)=C(1,1,1)+L*K-FN R(1): LET
C(K,1,2)=C(K-1,1,2)-FN R(1):
NEXT K
50 FOR J=1 TO S: FOR K=1 TO S
-J+1
60 LET C(K,G,1)=FN R(1)+(C(K,
F,1)+C(K+1,F,1))/2
70 LET C(K,G,2)=FN R(1)+H+(C(
K,F,2)+C(K+1,F,2))/2
80 PLOT C(K,F,1),C(K,F,2):
DRAW C(K+1,F,1)-PEEK 23677,C(
K+1,F,2)-PEEK 23678
90 DRAW C(K,G,1)-PEEK 23677,C
(K,G,2)-PEEK 23678: DRAW C(K,
F,1)-PEEK 23677,C(K,F,2)-PEEK
 23678
100 NEXT K: LET F=3-F: LET G=3
-F: NEXT J
110 FOR y=40 TO 0 STEP -.75
120 PLOT 0,y
130 FOR n=1 TO 100
140 LET a=RND*10
150 LET b=5-RND*10
160 IF a+PEEK 23677>255 THEN
LET n=100: DRAW 255-PEEK 23677
,b: GOTO 190
170 IF (PEEK 23678)+b<0 THEN
GOTO 150
180 DRAW a,b
190 NEXT n
200 NEXT y
210 FOR m=USR "a" TO USR "a"+7
: READ a: POKE m,a: NEXT m
220 DATA 16,56,84,16,56,84,146
,16
230 FOR n=1 TO 80
240 PRINT AT 17+INT (RND*4),
RND*31; BRIGHT 1; PAPER 4;
INVERSE 1;CHR$ 144;
250 NEXT n
260 PRINT #1; INVERSE 1;AT 0,4
;" RES=";S;"   RUGOSIDADE=";RG
;" "
270 GOTO 270
```

```
10 PMODE 4,1:PCLS:SCREEN 1,1
20 DIM C(200,1,1):F=0:G=1:C(0,0
,1)=150:C(0,0,0)=10
30 S=80:L=230/S:H=L/SQR(3):DEFF
NR(X)=3-RND(0)*6
40 FOR K=1 TO S:C(K,0,0)=C(0,0,
0)+L*K-FNR(0):C(K,0,1)=C(K-1,0,
1)-FNR(0):NEXT
50 FOR J=1 TO S:FOR K=0 TO S-J
60 C(K,G,0)=FNR(0)+(C(K,F,0)+C(
K+1,F,0))/2
70 C(K,G,1)=FNR(0)-H+(C(K,F,1)+
C(K+1,F,1))/2
80 LINE (C(K,F,0),C(K,F,1))-(C(
K+1,F,0),C(K+1,F,1)),PSET
90 LINE -(C(K,G,0),C(K,G,1)),PS
ET:LINE-(C(K,F,0),C(K,F,1)),PSE
T
100 NEXT:F=1-F:G=1-F:NEXT
110 GOTO 110
```

```
10  HGR2
20  DIM C(200,1,1):F = 0:G = 1:
C(0,0,1) = 150:C(0,0,0) = 10
30 S = 80:L = 230 / S:H = L /
SQR (3): DEF  FN R(X) = 3 -  RN
D (1) * 6
40  FOR K = 1 TO S:C(K,0,0) = C
(0,0,0) + L * K -  FN R(0):C(K,
0,1) = C(K - 1,0,1) -  FN R(0):
 NEXT
50  FOR J = 1 TO S: FOR K = 0 T
O S - J
60 C(K,G,0) =  FN R(0) + (C(K,F
,0) + C(K + 1,F,0)) / 2
70 C(K,G,1) =  FN R(0) - H + (C
(K,F,1) + C(K + 1,F,1)) / 2
80  HPLOT C(K,F,0),C(K,F,1) TO
C(K + 1,F,0),C(K + 1,F,1)
90  HPLOT  TO C(K,G,0),C(K,G,1)
: HPLOT  TO C(K,F,0),C(K,F,1)
100  NEXT :F = 1 - F:G = 1 - F:
 NEXT
110  GOTO 110
```

```
10 SCREEN 2
20 DIM C(200,1,1):F=0:G=1:C(0,0
,1)=150:C(0,0,0)=10
30 S=80:L=230/S:H=L/SQR(3):DEFF
NR(X)=3-RND(1)*6
40 FOR K=1 TO S:C(K,0,0)=C(0,0,
0)+L*K-FNR(0):C(K,0,1)=C(K-1,0,
1)-FNR(0):NEXT
50 FOR J=1 TO S:FOR K=0 TO S-J
60 C(K,G,0)=FNR(0)+(C(K,F,0)+C(
K+1,F,0))/2
70 C(K,G,1)=FNR(0)-H+(C(K,F,1)+
C(K+1,F,1))/2
80 LINE(C(K,F,0),C(K,F,1))-(C(K
+1,F,0),C(K+1,F,1))
90 LINE-(C(K,G,0),C(K,G,1)):LIN
E-(C(K,F,0),C(K,F,1))
100 NEXT:F=1-F:G=1-F:NEXT
110 GOTO 110
```



A linha 30 define o fator de escala dos triângulos e especifica o comprimento e a altura de um lado. A linha 40 preenche duas matrizes com as coordenadas iniciais de cada triângulo. Observe que há um fator aleatório — portanto, os valores irão variar em um pequeno intervalo a cada execução do programa. A linha 50 inicializa dois laços: um para desenhar os triângulos horizontalmente na tela, e outro para colocá-los mais acima.

O vértice de cada triângulo é especificado nas linhas 60 (coordenada X) e 70 (coordenada Y). A linha 80 move o cursor para o canto esquerdo do triân-

gulo e desenha sua base. No programa para o Spectrum, os **PEEK** subtraídos das coordenadas asseguram que os pontos não caiam fora da tela, o que provocaria uma interrupção do programa ao se tentar executar o **DRAW**. Na linha 90, o **DRAW** é direcionado para o vértice do triângulo e depois para o canto esquerdo (o início). A linha 100 completa o primeiro laço. Este desenha uma fileira de triângulos e, após redefinir algumas variáveis, de maneira a deslocar a fileira um pouco mais para cima, recomeça a construção da montanha. Na versão para o Spectrum, o programa completa a figura desenhando "árvores" na base da montanha.

FORMAS VARIADAS

Modificando os valores de algumas variáveis, você poderá obter muitas imagens diferentes. Mas, lembre-se, essa variação atinge apenas o tamanho, a posição e outros detalhes, mas não a forma geral da figura.

O próximo programa permite que você introduza uma forma inicial, construindo, a partir dela, uma figura fractal. O grande número de sub-rotinas recursivas pode ser um problema no Apple — esse micro só admite que uma sub-rotina seja chamada por ela mesma 24 vezes. Por isso deve-se ter cuidado ao escolher o nível de recursão.

```
10 POKE 23658,8: LET F=0: LET
A$="": LET CL=1
20 DIM X(100): DIM Y(100):
DIM T(30): DIM U(30): DIM V(
60): DIM W(60): DIM J(100)
25 BORDER 0: PAPER 7: INK 0:
CLS
30 GOSUB 140
40 GOSUB 350
50 INPUT "No DE NIVEIS DE REC
URSAO ";NR: IF ABS (INT (NR))
<1 THEN  GOTO 50
60 LET F=1: LET N=0: CLS
70 PLOT  INVERSE 1; OVER 1;
INT (127+X(P)),INT (85+Y(P))
80 GOSUB 500: IF P>=2 THEN
GOTO 80
90 LET A$=INKEY$: IF A$=""
THEN  GOTO 90
100 PRINT #1;"S PARA SAIR, QUA
LQUER OUTRA P/  CONTINUAR"
110 LET A$=INKEY$: IF A$=""
THEN  GOTO 110
120 IF A$<>"S" THEN  GOTO 25
130 CLS : STOP
140 IF F=0 THEN  GOTO 170
150 PRINT "MESMA FORMA INICIAL
(S/N) ? ";
160 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 160
165 PRINT A$
170 IF F=0 OR A$="N" THEN
GOSUB 230
180 FOR K=2 TO CV+1
190 LET P=K-1
200 LET X(P)=T(K): LET Y(P)=U(
K)
210 NEXT K
220 RETURN
230 INPUT "No DE VERTICES ";VT
240 FOR L=2 TO VT+1
250 INPUT "VERTICE ";(L-1);"="
'X,Y
260 LET T(L)=X*85: LET U(L)=Y*
85
270 IF L=2 THEN  PLOT INT (127
+T(L)),INT (85+U(L))
275 IF L<>2 THEN  DRAW 127+T(L
)-PEEK 23677,85+U(L)-PEEK
23678
280 NEXT L
290 PRINT "CURVA FECHADA (S/N)
? ";
300 LET A$=INKEY$: IF A$<>"N"
AND A$<>"S" THEN  GOTO 300
305 PRINT A$
310 IF A$="N" THEN  LET CV=VT:
PAUSE 0: RETURN
320 LET CV=VT+1: LET T(CV+1)=T
(2): LET U(CV+1)=U(2)
330 DRAW 127+T(CV)-PEEK 23677,
85+U(CV)-PEEK 23678
340 RETURN
350 CLS : IF F=0 THEN  GOTO
 380
360 PRINT "MESMO GERADOR (S/N)
? ";
370 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 370
375 PRINT A$
380 IF F=0 OR A$="N" THEN
GOSUB 400
390 RETURN
400 INPUT "No DE VERTICES DO G
ERADOR NAO   INCLUINDO AS EXTR
EMIDADES (0,0) E (1,0) ";GN
420 PLOT  INVERSE 1: OVER 1;85
,85
430 FOR M=2 TO GN+1
440 INPUT "VERTICE ";(M-1);"="
'X,Y
450 IF ABS (INT (X))>1 OR ABS
(INT (Y))>1 THEN  GOTO 440
460 LET V(M)=X: LET W(M)=Y:
LET X=X*85+85: LET Y=85+Y*85:
DRAW X-PEEK 23677,Y-PEEK 23678
470 NEXT M
480 DRAW 175-PEEK 23677,85-
PEEK 23678: PAUSE 0
490 RETURN
500 IF NR=N THEN  GOSUB 520
505 IF NR<>N THEN  GOSUB 570
510 RETURN
520 FOR W=1 TO GN+1
530 LET P=P-1
540 IF ABS X(P)>127 OR ABS Y(P
)>85 THEN  GOTO 560
550 DRAW 127+X(P)-PEEK 23677,
85+Y(P)-PEEK 23678
560 NEXT W: RETURN
570 LET N=N+1
580 IF N=1 THEN  LET JM=CV-1
585 IF N<>1 THEN  LET JM=GN+1
590 FOR E=1 TO JM
595 IF P=1 THEN  LET E=JM:
NEXT E: RETURN
600 LET TX=X(P): LET TY=Y(P)
610 LET BX=X(P-1): LET BY=Y(P-
1)
620 LET DX=TX-BX: LET DY=TY-BY
630 FOR F=2 TO GN+1
640 LET X(P)=DX*V(F)-DY*W(F)+
BX
650 LET Y(P)=DY*V(F)+DX*W(F)+
BY
660 LET P=P+1
670 NEXT F
680 LET X(P)=TX: LET Y(P)=TY
690 LET J(CL)=E: LET CL=CL+1:
GOSUB 500: LET CL=CL-1: LET E=
J(CL)
700 NEXT E
710 LET N=N-1
720 RETURN
```

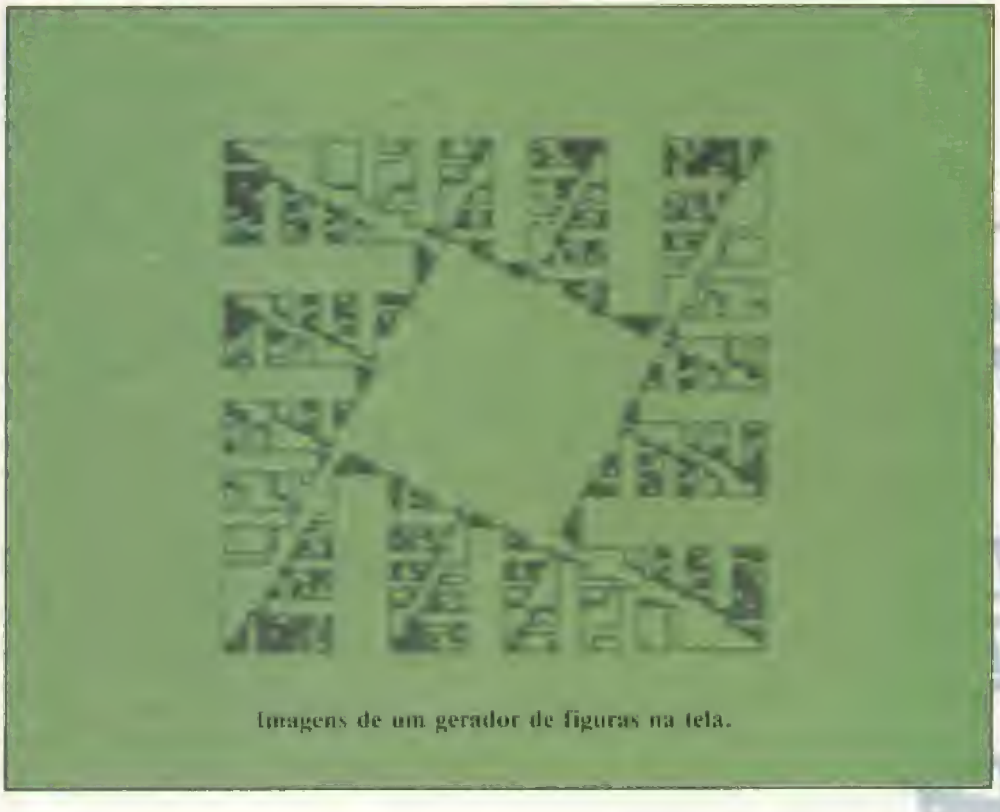
```
10 DIM X(50),Y(50),XT(50),YT(10
),XG(20),YG(20),J(50)
20 PMODE 4,1:COLOR 0,1:PCLS:CLS
30 GOSUB 140
40 GOSUB 350
50 INPUT"NO.DE NIVEIS DE RECURS
AO -";NR:NR=INT(NR):IF NR<1 THE
N 50
60 F=1:N=0:PCLS:SCREEN 1,0
70 LINE -(127+X(P),96-Y(P)),PRE
SET
80 GOSUB 500:IF P>0 THEN 80
90 A$=INKEY$:IF A$="" THEN 90
100 CLS:PRINT"S PARA SAIR, QUAL
QUER OUTRA     TECLA PARA CONTI
NUAR"
110 A$=INKEY$:IF A$="" THEN 110
120 IF A$<>"S" THEN 20
130 CLS:END
140 IF F=0 THEN 170
150 PRINT"MESMA FORMA INICIAL (
S/N)?";
160 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 160 ELSE PRINT AT A$
170 IF F=0 OR A$="N" GOSUB 230
180 FOR K=1 TO CV
190 P=K-1
200 X(P)=XT(K):Y(P)=YT(K)
210 NEXT
220 RETURN
230 INPUT"NO.DE VERTICES NA FOR
ME INICIAL ";VT:VT=INT(VT):IF V
T<1 THEN 230
240 FOR L=1 TO VT
250 PRINT"VERTICE ";L;:INPUT" -
 ";X,Y:IF ABS(X)>1 OR ABS(Y)>1
THEN 250
260 XT(L)=X*95:YT(L)=Y*95
270 IF L=1 THEN LINE -(127+XT(L
),96-YT(L)),PRESET ELSE LINE -(
127+XT(L),96-YT(L)),PSET
280 NEXT
290 PRINT"CURVA FECHADA (S/N)?";
300 A$=INKEY$:IF A$<>"N" AND A$
<>"S" THEN 300 ELSE PRINT A$
310 IF A$="N" THENCV=VT:GOSUB 7
30:RETURN
320 CV=VT+1:XT(CV)=XT(1):YT(CV)
=YT(1)
```

```
330 LINE-(127+XT(CV),96-YT(CV))
,PSET:GOSUB 730
340 RETURN
350 PCLS:IF F=0 THEN 380
360 PRINT"MESMO GERADOR (S/N) ?
";
370 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 370 ELSE PRINT A$
380 IF F=0 OR A$="N" GOSUB 400
390 RETURN
400 PRINT"NO.DE VERTICES NO GER
ADOR"
410 INPUT"NAO INCLUINDO AS EXTR
EMIDADES   (0,0) E (1,0) - ";GN
:GN=INT(GN):IF GN<1 THEN 410
420 DRAW"BM80,96"
430 FOR M=1 TO GN
440 PRINT "VERTICE ";M;"DO GERA
DOR ";:INPUT "- ";X,Y
450 IF ABS(X)>1 OR ABS(Y)>1 THE
N 440
460 XG(M)=X:YG(M)=Y:X=X*95+80:Y
=96-Y*95:LINE-(X,Y),PSET
470 NEXT
480 LINE-(175,96),PSET:GOSUB 73
0
490 RETURN
500 IF NR=N GOSUB 520 ELSE GOSU
B 570
510 RETURN
520 FOR L=1 TO GN+1
530 P=P-1
540 IF ABS(X(P))>127 OR ABS(Y(P
))>95 THEN 560
550 LINE-(127+X(P),96-Y(P)),PSE
T
560 NEXT:RETURN
570 N=N+1
580 IF N=1 THEN JM=CV-1 ELSE JM
=GN+1
590 FOR J=1 TO JM
600 TX=X(P):TY=Y(P)
610 BX=X(P-1):BY=Y(P-1)
620 DX=TX-BX:DY=TY-BY
630 FOR E=1 TO GN
640 X(P)=DX*XG(E)-DY*YG(E)+BX
650 Y(P)=DY*XG(E)+DX*YG(E)+BY
660 P=P+1
670 NEXT
680 X(P)=TX:Y(P)=TY
690 J(CL)=J:CL=CL+1:GOSUB 500:C
L=CL-1:J=J(CL)
700 NEXT J
710 N=N-1
720 RETURN
730 A$=INKEY$:SCREEN 1,0:K=1000
740 K=K-1:IF K>0 AND INKEY$=""
THEN 740
750 RETURN
```

```
5  HOME
10  DIM X(50),Y(50),XT(10),YT(1
0),XG(20),YG(20),J(50)
30  GOSUB 140
40  GOSUB 350
50  TEXT : HOME : INPUT "QUANTO
S NIVEIS DE RECURSAO ";NR:NR =
 INT (NR): IF NR < 1 THEN 50
60 F = 1:N = 0: HGR2
70  HCOLOR= 3: HPLOT 127 + X(P)
,96 - Y(P)
80  GOSUB 500: IF P > 0 THEN 80
90  GET A$
100  TEXT : HOME : PRINT "<S> P
ARA SAIR E QUALQUER OUTRA PARA
CON-TINUAR"
110  GET A$
120  IF A$ <  > "S" THEN 30
130  HOME : END
140  IF F = 0 THEN 170
150  TEXT : HOME : PRINT "MESMA
 FORMA INICIAL?(S/N)";
160  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 160
165  PRINT A$
170  IF F = 0 OR A$ = "N" THEN
 GOSUB 230
180  FOR K = 1 TO CV
190 P = K - 1
200 X(P) = XT(K):Y(P) = YT(K)
210  NEXT
220  RETURN
230  INPUT "QTOS. VERTICES NA F
ORMA INICIAL ";VT:VT =  INT (VT
): IF VT < 1 THEN 230
240  FOR L = 1 TO VT
250  PRINT "VERTICE ";L;: INPUT
 " - ";X,Y: IF  ABS (X) > 1 OR
 ABS (Y) > 1 THEN 250
260 XT(L) = X * 95:YT(L) = Y *
95
265  NEXT
270  PRINT "CURVA FECHADA (S/N)
 ?";
280  GET A$: IF A$ <  > "N" AND
 A$ <  > "S" THEN 280
285  PRINT A$: FOR T = 1 TO 300
: NEXT : HGR2
290  FOR L = 1 TO VT
300  IF L = 1 THEN  HCOLOR= 3:
HPLOT 127 + XT(L),96 - YT(L)
302  IF L <  > 1 THEN  HCOLOR=
3: HPLOT  TO 127 + XT(L),96 - Y
T(L)
307  NEXT
310  IF A$ = "N" THEN CV = VT:
GOSUB 730: RETURN
320 CV = VT + 1:XT(CV) = XT(1):
YT(CV) = YT(1)
330  HPLOT  TO 127 + XT(CV),96
- YT(CV): GOSUB 730
340  RETURN
350  IF F = 0 THEN 380
360  TEXT : HOME : PRINT "MESMO
 GERADOR?(S/N)";
370  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 370
375  PRINT A$
380  IF F = 0 OR A$ = "N" THEN
 GOSUB 400
390  RETURN
400  TEXT : HOME : INPUT "QUANT
OS VERTICES NO GERADOR NAO INCL
UIN-DO AS EXTREMIDADES (0,0) E
(1,0)  ";GN
410 GN =  INT (GN): IF GN < 1 T
HEN 400
430  FOR M = 1 TO GN
440  PRINT "VERTICE ";M;" DO GE
RADOR";: INPUT " - ";XG(M),YG(M
)
450  IF  ABS (XG(M)) > 1 OR  AB
S (YG(M)) > 1 THEN 440
455  NEXT
457  HGR2 : HCOLOR= 3: HPLOT 80
,96
458  FOR M = 1 TO GN
460 X = XG(M) * 95 + 80:Y = 96
- YG(M) * 95: HPLOT  TO X,Y
470  NEXT
480  HPLOT  TO 175,96: GOSUB 73
0
490  RETURN
500  IF NR = N THEN  GOSUB 520
505  IF NR <  > N THEN  GOSUB 5
70
510  RETURN
520  FOR L = 1 TO GN + 1
530 P = P - 1
540  IF  ABS (X(P)) > 127 OR  A
BS (Y(P)) > 95 THEN 560
550  HPLOT  TO 127 + X(P),96 -
Y(P)
560  NEXT : RETURN
570 N = N + 1
580  IF N = 1 THEN JM = CV - 1
585  IF N <  > 1 THEN JM = GN +
 1
590  FOR J = 1 TO JM
600 TX = X(P):TY = Y(P)
610 BX = X(P - 1):BY = Y(P - 1)

620 DX = TX - BX:DY = TY - BY
630  FOR E = 1 TO GN
640 X(P) = DX * XG(E) - DY * YG
(E) + BX
650 Y(P) = DY * XG(E) + DX * YG
(E) + BY
660 P = P + 1
670  NEXT E
680 X(P) = TX:Y(P) = TY
690 J(CL) = J:CL = CL + 1: GOSU
B 500:CL = CL - 1:J = J(CL)
700  NEXT J
710 N = N - 1
730  FOR T = 1 TO 1000: NEXT
740  RETURN
```

```
5 CLS
10 DIM X(50),Y(50),XT(10),YT(10
),XG(20),YG(20),J(50)
20 SCREEN 0
30 GOSUB 140
40 GOSUB 350
50 SCREEN 0:INPUT "QUANTOS NIVE
IS DE RECURSAO ";NR:NR=INT(NR):
IF NR<1 THEN 50
60 F=1:N=0:SCREEN 2
70 LINE-(127+X(P),96-Y(P)),4
80 GOSUB 500:IF P>0 THEN 80
90 A$=INKEY$:IF A$="" THEN 90
100 SCREEN 0:PRINT"<S> PARA SAI
R E QUALQUER OUTRA PARA CONTINU
AR"
110 A$=INKEY$:IF A$="" THEN 110
120 IF A$<>"S" THEN 20
130 CLS:END
140 IF F=0 THEN 170
150 SCREEN 0:PRINT"MESMA FORMA
INICIAL?(S/N)";
160 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 160 ELSE PRINT A$
170 IF F=0 OR A$="N" THEN GOSUB
 230
180 FOR K=1 TO CV
```



Imagens de um gerador de figuras na tela.

```
190 P=K-1
200 X(P)=XT(K):Y(P)=YT(K)
210 NEXT
220 RETURN
230 INPUT "QTOS. VERTICES NA FO
RMA INICIAL";VT:VT=INT(VT):IF V
T<1 THEN 230
240 FOR L=1 TO VT
250 PRINT "VERTICE ";L;:INPUT"
- ";X,Y:IF ABS(X)>1 OR ABS(Y)>1
 THEN 250
260 XT(L)=X*95:YT(L)=Y*95
265 NEXT
270 PRINT"CURVA FECHADA (S/N) ?
 ";
280 AS=INKEYS:IFAS<>"N" AND AS<
>"S" THEN 280 ELSE PRINT AS:FOR
 T=1 TO 300:NEXT
290 SCREEN 2:FOR L=1 TO VT
300 IF L=1 THEN LINE -(127+XT(L
),96-YT(L)),4 ELSE LINE -(127+X
T(L),96-YT(L)),11
305 NEXT
310 IF AS="N" THEN CV=VT:GOSUB
730:RETURN
320 CV=VT+1:XT(CV)=XT(1):YT(CV)
=YT(1)
330 LINE -(127+XT(CV),96-YT(CV)
),11:GOSUB 730
340 RETURN
350 IF F=0 THEN 380
360 SCREEN 0:PRINT"MESMO GERADO
R?(S/N)";
370 AS=INKEYS:IF AS<>"S" AND AS
<>"N" THEN 370 ELSE PRINT AS
380 IF F=0 OR AS="N" THEN GOSUB
 400
390 RETURN
400 INPUT"QUANTOS VERTICES NO G
ERADOR NAO INCLUINDO AS EXTREMI
DADES (0,0) E (1,0)";GN
410 GN=INT(GN):IF GN<1 THEN 400
430 FOR M=1 TO GN
440 PRINT"VERTICE ";M;" DO GERA
DOR";:INPUT" - ";XG(M),YG(M)
450 IF ABS(X)>1 OR ABS(Y)>1 THE
N 440
455 NEXT
457 SCREEN 2:PSET(80,96),11
458 FOR M=1 TO GN
460 X=XG(M)*95+80:Y=96-YG(M)*95
:LINE-(X,Y),11
470 NEXT
480 LINE-(175,96),11:GOSUB 730
490 RETURN
500 IF NR=N THEN GOSUB 520 ELSE
 GOSUB 570
510 RETURN
520 FOR L=1 TO GN+1
530 P=P-1
540 IF ABS(X(P))>127 OR ABS(Y(P
))>95 THEN 560
550 LINE-(127+X(P),96-Y(P)),11
560 NEXT:RETURN
570 N=N+1
580 IF N=1 THEN JM=CV-1 ELSE JM
=GN+1
590 FOR J=1 TO JM
600 TX=X(P):TY=Y(P)
610 BX=X(P-1):BY=Y(P-1)
620 DX=TX-BX:DY=TY-BY
630 FOR E=1 TO GN
640 X(P)=DX*XG(E)-DY*YG(E)+BX
650 Y(P)=DY*XG(E)+DX*YG(E)+BY
660 P=P+1
670 NEXT
680 X(P)=TX:Y(P)=TY
690 J(CL)=J:CL=CL+1:GOSUB 500:C
L=CL-1:J=J(CL)
700 NEXT J
710 N=N-1
720 RETURN
730 AS=INKEYS:K=1000
740 K=K-1:IF K>0 AND INKEYS=""
THEN 740
750 RETURN
```

Quando você executa o programa, a linha 230 pede o número de vértices do desenho inicial que irá gerar a figura fractal. Convém traçar essa figura previamente em uma folha de papel. Marque dois pontos representando o início e o fim da linha e una-os por meio de pequenas retas. Conte o número de cantos e forneça esse dado ao computador. Não exagere no número de retas: lembre-se que você terá que especificar as coordenadas de cada vértice (linha 250). Estas devem ter valores compreendidos entre -1 e 1.

O laço entre as linhas 240 e 280 (240 e 305 nas versões para o MSX e o Apple) permite que você introduza as coordenadas e desenhe a forma inicial, determinando inclusive se sua figura será fechada ou aberta (linha 290).

Em seguida (linhas 400 a 490), o computador solicita ao usuário a definição da figura que substituirá cada linha reta do desenho inicial — essa figura é geralmente chamada de gerador. Como você fez para a forma inicial, desenhe a figura e passe a informação para o computador.

Finalmente, você deverá especificar o número de níveis de recursão. Quando esse valor é digitado, a linha 80 chama a sub-rotina que verifica se o programa está rodando pela primeira vez. Em caso afirmativo, o programa é desviado para a rotina principal (linha 570 à 720), que desenha a figura fractal. Se o programa já estava sendo rodado, o computador dá ao usuário a chance de redefinir o gerador.

TESTE

Para experimentar o programa, digite o valor 3 para o número de vértices da forma inicial. Em seguida, introduza as coordenadas -0.5 e -0.2 para o vértice 1; 0 e 0.4 para o vértice 2; 0.5 e -0.2 para o vértice 3. Ao responder N à pergunta "CURVA FECHADA?", aparecerá na tela um triângulo sem base. Como foi feito para a forma inicial, entre 3 para o número de vértices do gerador e defina as coordenadas: 0.2 e 0 para o vértice 1; 0.4 e 0.8 para o vértice 2; 0.6 e 0 para o vértice 3. Esses dados definem a imagem de um triângulo sem base sobre uma linha. Por fim, introduza o valor 5 para o nível de recursão (3 para o Apple) e observe a geração da figura fractal.

BITS E BYTES EM BASIC

A manipulação dos bits individuais de um byte de memória por meio de comandos do BASIC parece uma tarefa impossível. Mas não é: aprenda os truques de programação necessários para isso.

O byte é a unidade básica de memória usada em microcomputadores. Os diversos comandos da linguagem BASIC tratam o byte como a unidade mínima de informação: os caracteres individuais de uma cadeia alfanumérica, por exemplo, podem ser isolados e processados individualmente com comandos do tipo **MID$**, **LEFT$**, **RIGHT$**, **LEFT$$**, **CHR$**, **ASC** etc. Cada caractere ocupa um byte. Os valores numéricos, por sua vez, em geral ocupam dois ou mais bytes, dependendo de sua precisão (sobre a armazenagem de números, veja o artigo da página 894). Usando os comandos **VARPTR**, **PEEK** e **POKE**, temos acesso individual aos bytes onde variáveis, vídeo, teclado etc. estão armazenados.

O BIT

Existe, porém, uma unidade de informação menor que o byte. Essa unidade é o bit (*binary digit*, ou dígito binário, em inglês), que corresponde aos dígitos 0 e 1 que todo computador digital usa como base de representação numérica. Cada byte tem oito bits, numerados de 0 (o bit menos significativo) a 7 (o bit mais significativo).

Muitas vezes, surge a necessidade de manipular diretamente os bits da memória. Para isso, convém utilizar uma rotina em linguagem de máquina, que tem comandos específicos para o acesso e a manipulação de bits individuais e é mais rápida que qualquer rotina em linguagem de alto nível.

Entretanto, poucos sabem que é perfeitamente possível realizar manipulações diretas de bits com os comandos já existentes no BASIC. Não é tão rápido nem tão direto quanto em linguagem de máquina, mas funciona bem.

Neste artigo, você aprenderá vários truques que permitem a manipulação de bits em quase todos os microcomputadores. Os comandos do BASIC mais relevantes para esse fim são o **CHR$**, o **ASC** (ou **CODE**, no ZX-81) e os operadores lógicos **AND**, **NOT** e **OR**.

Com as funções **CHR$** e **POKE**, podemos criar bytes com uma determinada configuração de bits. **CHR$** é utilizada no trabalho com variáveis simbólicas dentro de um programa em BASIC; **POKE** nos dá acesso direto às locações absolutas da memória RAM.

CHR$(1), por exemplo, define um byte com o bit menos significativo igualado a 1, e todos os outros igualados a 0: 0000001 em binário. **CHR$(2)** gera um byte com os dois bits menos significativos igualados a 1: 00000011. **CHR$(3)** define o byte 00000010 e assim por diante. Com o auxílio de uma tabela de correspondência entre números decimais de 0 a 255 e os binários equivalentes, pode-se produzir qualquer padrão de bits ligados e desligados.

Para converter o padrão de bits em um número decimal, devemos considerar cada bit como o índice de uma potência de 2 e somar o resultado. Suponhamos um byte com os bits 0, 3 e 4 ligados:

```
00001101
```

O número decimal correspondente é:

$$2^4 + 2^3 + 2^0 = 16 + 8 + 1 = 25$$

Eis algumas expressões úteis para ligar ou desligar só o bit B de um byte armazenado em um caractere **B$**:

Para ligar:

```
B$=CHR$(CODE(B$) OR 2**B)
```

Para desligar:

```
B$=CHR$(CODE(B$) AND NOT 2**B)
```

Para ligar:

```
B$=CHR$(ASC(B$) OR 2**B)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2**B)
```

Para ligar:

```
B$=CHR$(ASC(B$) OR 2↑B%)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2↑B%)
```

Para ligar:

```
B$=CHR$(ASC(B$) OR 2^B%)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2^B%)
```

Observe que, com exceção dos micros da linha Sinclair, que não têm variáveis inteiras, usamos a notação **B%** para indicar que esse valor deve ser inteiro. Isso ajuda a evitar erros de cálculo, caso **B** seja um bit real de precisão simples.

Para tornar o bit B de um byte igual a 1, usamos o operador lógico **OR**. Suponhamos que você queira igualar a 1 o bit número 7 (o mais significativo) de um byte igual a 00001011. Teremos, então, a operação, em decimal:

$$25 \textbf{ OR } 2^7 = 25 \textbf{ OR } 128$$

o que, em binário, dá:

```
00001011  OR
10000000
--------
10001011
```

Note que **OR** tem o efeito de operar individualmente sobre cada par de bits na mesma posição, nos operandos, de acordo com o seguinte esquema:

```
0 OR 0  =  0
1 OR 0  =  1
0 OR 1  =  1
1 OR 1  =  1
```

Para colocar 0 em um bit qualquer do byte **B$**, usamos a operação **AND NOT**. De fato:

```
0 AND NOT 0 = 0 AND 1 = 0
1 AND NOT 0 = 1 AND 1 = 1
0 AND NOT 1 = 0 AND 0 = 0
1 AND NOT 1 = 1 AND 0 = 0
```

pois a operação **NOT** transforma um bit 0 em 1 e vice-versa, equivalendo, portanto, a uma *inversão* do bit.

Se você quisesse transformar o biná-

rio 10001011 em 00001011, igualando o bit 7 a 0, teria:

```
     10001011 AND
NOT  10000000
```

que é a mesma coisa que:

```
10001011 AND
01111111
---------
00001011
```

Pode ser que, em vez de trabalhar com bytes armazenados em caracteres literais, você queira manipular os bits individuais de um número inteiro — que, na maioria dos micros, é armazenado em dois bytes contíguos, em um total de dezesseis bits. Nesse caso, use as expressões apresentadas a seguir. Mas lembre-se de que alguns computadores, como o ZX-81, podem armazenar de forma diferente um número inteiro.

Para ligar:

```
I% = I% OR 2↑B%
```

Para desligar:

```
I% = I% AND NOT 2↑B%
```

Para ligar:

```
I% = I% OR 2^B%
```

Para desligar:

```
I% = I% AND NOT 2^B%
```

Essas expressões funcionam com um conjunto de dezesseis bits — ou seja, se quisermos ligar o bit 13 do byte I%, faremos a operação:

```
I% = I% OR 2 ^ 12
```

ou

```
I% = I% OR 8192%
```

A expressão **AND NOT** desliga o B-ésimo bit de I%.

Os micros ZX-81 e Spectrum não podem utilizar essas expressões, já que não têm variáveis inteiras. Todos os outros micros podem, pois trabalham com a versão microsoft do BASIC.

COMBINAÇÃO DE BITS

É possível ligar uma combinação de bits em uma variável literal ou inteira. Para isso, usa-se uma expressão de soma dos expoentes de 2 correspondentes a cada bit a ser ligado, combinados por operações **OR**. Para ligar os bits 3 e 5 do byte em **T$**, faça:

```
T$ = CHR$((2**3) OR (2**5))
```

```
T$ = CHR$((2↑3) OR (2↑5))
```

```
T$ = CHR$((2^3) OR (2^5))
```

TESTANDO OS BITS

Da mesma forma que podemos ligar ou desligar bits individuais em linguagem BASIC, é bastante fácil examinar o valor desses bits, através de uma expressão simples:

```
R = CODE (B$) AND 2**B
```

```
R = ASC (B$) AND 2**B
```

```
R% = ASC(B$) AND 2↑B%
```

ou

```
R% = I% AND 2↑B%
```

```
R% = ASC(B$) AND 2^B%
```

ou

```
R% = I% AND I^B%
```

A operação **AND** isola apenas o valor do bit mascarado pelo byte igual a 2^B. Por exemplo: qual é o valor do bit número 6 do byte 01001011?

```
01001011  AND
01000000
---------
01000000
```

que equivale a 89 **AND** 64, cujo resultado é 64 em decimal.

Como o número resultante dessa operação é maior do que 0, o resultado é verdadeiro. Se fosse igual a 0, teríamos então um resultado falso.

Para transformar o resultado **R** em valor 1, fazemos:

```
R% = ABS(R%>(2 ↑ B% - 1))
```

Em nosso exemplo, isso daria:

```
ABS (64>63) = 1
```

Se R% fosse 0, teríamos:

```
ABS (0>63) = 0
```

Eis aqui um programa para testar todos os bits de um byte de entrada:

```
110 PRINT "ENTRE UM NUMERO EN-
TRE 0 E 255)
120 INPUT N
130 LET N$=CHR$(N)
140 FOR I=0 TO 7
150 PRINT "BIT ";I;" = ";
160 LET C=ASC(N$) AND 2**I
170 IF C THEN PRINT "SIM"
180 IF NOT C THEN PRINT "NAO"
190 NEXT I
```

O programa para o ZX-81 é o mesmo do Spectrum, com esta alteração:

```
160 LET C=CODE(N$) AND 2**I
```

```
110 PRINT "ENTRE UM NUMERO EN-
TRE 0 E 255)
120 INPUT N%
130 N$=CHR$(N%)
140 FOR I%=0 TO 7
150 PRINT "BIT ";I%;" = ";
160 C%=ASC(N$) AND 2↑I%
170 IF C% THEN PRINT "SIM"
180 IF NOT C% THEN PRINT "NAO"
190 NEXT I%
```

O programa para o MSX e o Apple é igual ao anterior, com a modificação:

```
160 C%=ASC(N$) AND 2^I%
```

MAPAS DE BITS

Até agora, vimos como manipular os bits de um único byte ou conjunto de dois bytes. Como uma cadeia alfanumérica contém até 255 caracteres, podemos manipular um conjunto muito maior de bits (um máximo de 8 × 255, ou 2040 bits). Esse conjunto é chamado de *cadeia de mapeamento de bits (bit-mapstring)*, e tem muitas aplicações em jogos, bancos de dados e outros programas de caráter profissional.

Por exemplo, é possível armazenar em uma cadeia desse tipo condições sim/não ou verdadeiro/falso. Se um bit em determinada posição dessa cadeia estiver ligado, temos uma condição sim, ou verdadeira; se estiver desligado, uma condição não, ou falsa.

O comprimento da cadeia dependerá do número de condições (*flag bits* ou bits indicadores) que queremos incluir. Se o número de condições for **N**, o número de bytes da cadeia será:

```
L = INT(N/B)+1
```

Para inicializar um mapa **M$**, zeramos todos os bits:

```
45 CLS
50 INPUT "NUMERO DE BITS NA CAD
EIA ";n
60 IF n>255 THEN GOTO 50
70 FOR i=1 TO INT(n/8)+1
80 LET m$=m$+CHR$(0)
90 NEXT i
```

```
45 CLS
50 INPUT "NUMERO DE BITS NA CAD
EIA ";N
60 IF N>255 THEN 50
70 M$=STRING$(INT(N/8)+1,0)
```

Para ligar, desligar e testar um bit qualquer do conjunto total de **N**, precisaremos de algumas funções poderosas: **FNL$**, **FND$** e **FNT$**.

Essas funções são bastante complexas, mas muito rápidas. Procure analisá-las passo a passo para entender o que elas executam:

```
20 DEF FNL$(B)=m$(TO INT(b/8))
+CHR$(ASC(m$(INT(b/8)+1)) OR 2
**(b-INT(b/8)*8))+m$(INT(b/8)+2
 TO)
30 DEF FND$(b)=m$(TO INT(b/8))
+CHR$(ASC(m$(INT(B/8)+1)) AND
NOT 2**(b-INT(b/8)*8))+m$(INT
b/8)+2 TO)
40 DEF FNT(b)=ABS((ASC(m$(INT
(b/8)+1)) AND 2**(b-INT(b/8)*
8))<>0)
```

```
20 DEF FNL$(B)=LEFT$(M$,INT(B/8
))+CHR$(ASC(MID$(M$,INT(B/8)+1,
1)) OR 2↑(B-INT(B/8)*8))+MID$(M
$,INT(B/8)+2)
30 DEF FND$(B)=LEFT$(M$,INT(B/8
))+CHR$(ASC(MID$(M$,INT(B/8)+1,
1)) AND NOT 2↑(B-INT(B/8)*8))+M
ID$(M$,INT(B/8)+2)
40 DEF FNT(B)=ABS((ASC(MID$(M$,
INT(B/8)+1)) AND 2↑(B-INT(B/8)*
8))<>0)
```

Por fim, acrescentamos o restante do programa de demonstração das funções de manipulação e consulta de bits:

```
200 PRINT "PROGRAMA DE DEMONSTR
ACAO DE MAPAS DE BITS"
210 INPUT "(L)IGA (D)ESLIGA (T)
ESTA (F)IM ";OP$
230 IF OP$="F" THEN STOP
240 INPUT "NUMERO DO BIT (0 A "
;N;")";B
260 IF B>N THEN GOTO 240
270 IF OP$="L" THEN M$=FNL$(B)
280 IF OP$="D" THEN M$=FND$(B)
290 IF OP$="T" THEN PRINT "BIT
";B," = ";FNT(B)
310 GOTO 210
```

APLICAÇÕES

Você não terá dificuldade em encontrar aplicações para os truques explicados neste artigo.

Qual é a vantagem de empregar cadeias alfanuméricas, em vez de conjuntos numéricos, para armazenar grupos de bits?

Como vimos no artigo *Armazenagem de Programas* (página 1001), a cadeia alfanumérica é armazenada na memória do microcomputador na forma de uma seqüência contígua de bytes, cada qual ocupado por um caractere. O primeiro byte da seqüência sempre indica o número de caracteres da variável literal (*string*). Esse número é "lido" por intermédio da função **LEN**, não havendo, no BASIC (como ocorre em outras linguagens), um byte encarregado de indicar ao interpretador onde termina o string.

A vantagem de empregar cadeias alfanuméricas decorre justamente desse sistema de armazenagem usado pelo interpretador BASIC.

A disposição seqüencial dos bytes na memória facilita enormemente a programação de rotinas de acesso aos bits individuais de uma longa cadeia. Um string de trinta bytes, por exemplo, equivale a uma cadeia ininterrupta de 240 bits, que podem ser lidos ou modificados um a um, ou em grupos que se sobreponham aos limites entre bytes. Isso é muito mais difícil de ser feito quando a variável contém elementos separados, como é o caso dos conjuntos numéricos.

No desenvolvimento de jogos, por exemplo, a possibilidade de manipular bits amplia bastante as alternativas do programador. Entre outras coisas, você poderá modificar os gráficos impressos no vídeo alterando diretamente as locações de memória.

Os recursos de programação aqui examinados também serão úteis quando for necessário comprimir texto, em função de limitações da memória. Na série de artigos sobre o assunto, que iniciamos na página 1332, apresentamos diversos algoritmos interessantes para a redução do espaço ocupado por um texto em até 50%. A maioria desses algoritmos procura colocar dois ou mais códigos de caracteres em um único byte. Como exercício, tente implementar o supercompressor baseado na estatística de pares de letras em um texto (veja os artigos mencionados).

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Craft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

APLICAÇÕES
O que é um editor musical. Como digitar as notas. Execução de melodias no teclado. Composição de peças.

SOFTWARE
Processadores de texto. Hardware necessário. Impressão.

APLICAÇÕES
Projeto arquitetônico computadorizado: complemento das listagens para os diferentes micros. Instruções.

PROGRAMAÇÃO BASIC
Formatação de telas. Rotina para entrada de dados.

PROGRAMAÇÃO BASIC
Desenho em perspectiva. Ponto de fuga e ponto de vista. Redução. Sombreamento.